UNIVERSITY OF
LIVERPOOL

Visual Servoing of Compliant Welding Manipulators

Thesis submitted in accordance with the requirements of the
University *of* Liverpool for the
Degree of Doctor in Philosophy

James Ross Buckle

September 2007

Abstract

Established robot control technology, on the whole, relies on the assumption that the links between joints are fully rigid and therefore can be used to accurately and repeatably calculate where the end effector is in relation to the base, by calculation involving the angles of the joints and the length of the links. Assuming the links and drive mechanisms are rigid and accurately machined, this can be used to create a robot that is very accurate and highly repeatable - two key properties of manipulators. Accuracy can be described as a measure of the absolute error in manipulator position from the chosen target, repeatability as a measure of the spread of positions on successive iterations of the same process. A system can exhibit high repeatability but low accuracy; repeatedly reaching exactly the same location, but some distance from where it should be. A system cannot exhibit high accuracy and low repeatability; accuracy implies an ability to move close to a target on each attempt.

The advantage of robots with accurately machined, rigid links connected by high precision encoders and drive gears are the relatively simple calculations of kinematics and dynamics of the system. The disadvantages are:

- The financial expense of the precision machining of the links.
- The weight and size of the links required for lifting reasonable loads without link flex destroying the accuracy of the end effector position data.
- The cost of the joint motors and encoders - high precision, high torque motors are expensive.

When the robot cannot be manufactured as a rigid system, such as for use in tasks where low manipulator mass is crucial, the equations of motion of the robot are very complex and specific to that manipulator and payload. Methods of reducing the modeling required use large numbers of additional sensors, their signal processing systems add complexity and cost to the systems. These increased costs are justifiable in applications such as operations in space, however they counteract the savings that could be used to reduce the cost of general-purpose robots.

By the application of end-effector mounted computer vision, so called Eye-In-Hand (EIH), providing feedback control to the basic control system, flexibility in the

links can be tolerated and compensated for without modeling of the link flexibility. This allows the robot to be manufactured from cheaper links and drives, significantly reducing the cost of the robot. This may allow the introduction of robots into areas not currently viable for financial reasons such as low-value-adding processes where a robot would be considered useful but too expensive to justify. Existing methods of working around flexibility employ either fault-tolerant gripper technology or complex modeling and feedback from a large number of sensors. This thesis details the creation and testing of a single high speed EIH visual control system to compensate for unknown manipulator link flex. This is done with a view to assessing the feasibility of producing and controlling a compliant robot with minimal sensors or modeling, which would allow reduction in its manufacturing costs.

Acknowledgements

Thanks to my supervisor, Professor J.S. Smith, on two levels; for academic support, advice and guidance throughout the course of this work, but also as a friend, for interesting discussions of foreign affairs. Thanks to Professor S. Hall, for providing the facilities to allow me to complete this research at the University of Liverpool.

Thanks also to my friends; Kev, for keeping me sane and putting things into perspective, Paul, for support and feedback, Sonu, for eternal optimism and vocal accompaniment on journeys and Wes, for constant distractions with new and interesting project ideas.

Finally, I owe a great debt to those closest to me. To my parents, John and June, for your constant support, love and advice. Damon, for always being available to talk to, for your calmness (and kitesurfing) during the storm. To Lee, for instilling curiosity and determination - you are greatly missed. Emma, you understand me and always know how to make things right, thank-you for being you. For believing in me, even when I did not, this thesis is dedicated to you all, my family.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university, or institution of learning.

Copyright

Contents

Nomenclature

ω_n	Resonant Frequency
${}^0T_6$	Manipulator transformation matrix for a six DOF manipulator
A_i	Generalised link transformation matrix
k_d	Differential Gain
k_i	Integral Gain
k_p	Proportional Gain
L_n	Length of link n
T	Specific transformation matrix
AI	Artificial Intelligence
API	Application Programming Interface
BIOS	Basic Input/Output System
bps	bits per second
CCD	Charge-Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DH	Denavit and Hartenberg
DOF	Degrees of Freedom
EIH	Eye In Hand
FIFO	First In First Out
FL	Fuzzy Logic
FPGA	Field Programmable Gate Array
fps	Frames Per Second

HAL Hardware Abstraction Layer

IBVS Image Based Visual Servoing

IPC Inter-Process Communication

ISR Interrupt Service Routine

k Spring Constant

LXRT LinuX Real-Time

MBX Real-time Mailbox Structure

OS Operating System

P Proportional Control

PBVS Position Based Visual Servoing

PD Proportional-Derivative Control

PIC Programmable Interrupt Controller

PID Proportional-Integral-Derivative

PUMA Programmable Universal Manipulator Arm

RT Real-Time

RTAI Real-Time Application Interface

SDK Software Developers Kit

SHM RTAI real-time shared memory area

T Sample Time

VAL Variable Assembly Language

x,y,z Cartesian Coordinates

ZN Ziegler-Nichols

List of Figures

Chapter 1

Introduction

Robotics is an important field of research with rapid advancements in existing technology, as well as new concepts, presented frequently. A large base of literature presents significant improvements in theory, modeling, simulation and design of control systems and software. These advances are not as apparent as one would expect in the industrial world, or in the application of these ideas to practical experimentation and development.

In the field of manipulator systems, especially industrial manipulators, the large majority of existing technology relies on the assumption that the manipulator is rigid. Rigid manipulators make the calculation of end effector position relatively simple, using only joint encoders to determine each joint's relative extent of actuation. The control systems required to position and move such manipulators continue to be extensively researched and the limitations of them and their underlying rigid body assumption have been assessed recently by Middleton [1].

Most rigid manipulators have very high accuracy and repeatability, the PUMA 560 has a quoted repeatability of $\pm 0.1mm$ and a maximum end effector speed of $1ms^{-1}$. These are only achieved by way of the high precision of the gearing and links, careful control over the gear backlash and play in the joint bearings. If even

a small amount of pivot tolerance or flexibility were allowed to be present, the rigid body assumption collapses. Then the robot controller thinks it is aware of the end effector location, but is mistaken. This leads to process errors and possibly manipulator or workpiece damage in extreme situations. One such area is robotic welding - the act of following a prescribed weld path, in order to join two metal items. If a manipulator were used that had even minor compliance in the links, the operation would be undertaken faithfully according to rigid model assumptions, yet because the manipulator has some level of unpredictability the weld could be significantly off target. This flexibility and backlash can occur due to wear, such as that which becomes present in an industrial manipulator after many hours of operation, or could be due to faults or cost savings in the original construction.

In a serial robot, one in which each link is attached to the distal end of the previous forming a chain of links end to end, the errors in each link and drive are compounded and can result in very significant error, especially under heavy loads. If the flexibility of the manipulator is known, as well as the payload mass, the error can be predicted by modeling using one of several techniques and the errors negated.

Another method of negating such errors is to apply an array of sensor systems to the manipulator - such as multiple strain gauges per link - in order to observe the actual mode and magnitude of flex and feed that back into the control system. This vastly increases the sensor system complexity, modeling and control system workload. In order to assure these errors are not present, industrial manipulators are constructed with the highest of machining accuracy and joint gearing and bearings are adjusted to very fine tolerances.

Similarly the links are designed to be extremely rigid - Figure 1.1 and Figure 1.2 show the aluminium structure of the main links of the PUMA 560 for example - they are generally constructed as a strong box or tube cross section with what seem like excessive dimensions for their allowable payload. The manufacturer recommends drive backlash checking and adjustment every 42 days of constant use and bearing

play check every 80. This requires taking the robot offline to perform the tests and adjustments and is a good indication of the critical nature of the tolerances.

The payload limit for the PUMA 560 (Mk-II) is 2.5kg, payloads greater than this seriously degrade the quoted accuracy and repeatability of the system and can damage the drive mechanisms, especially on the smaller joints such as the wrist joints. The mass of the arm is around 65kg, configuration dependant, which appears disproportionate to the allowable payload, however, it does include the arm joint sensors, gearing and drive motors. The teardrop shape of the link structures is due to the manufacturer attempting to locate the heaviest components (motors, gearboxes) at the end nearest the point of rotation for that link, reducing the overall link inertia.

Rigid manipulators can be assumed to be simple control problems with a fixed number of possible degrees of freedom. The PUMA 560 in its original form has six degrees of freedom and, making the assumption that it is fully rigid, its end effector position can be calculated from only the known values of the joint encoders. The forward and inverse kinematic equations provide us with a direct conversion between joint angles and the tools position and orientation in world coordinates. Manipulator kinematics are calculated either using the Denavit and Hartenberg [2] coordinate frame assignments and a matrix algebra approach or, for simpler manipulators, geometric analysis.

The assumption that the manipulator is rigid is not necessarily accurate; even with a robot such as the PUMA, a series of small deformations in the manipulator, either local to the actuators or dispersed throughout the material of the manipulator, can lead to inaccuracies in the calculated position. For robots with low link deformation, this may present an acceptable accuracy loss. With links that exhibit large deformations, the accuracy loss can not be considered negligible and must either be removed by feedback sensing or predicted by modeling. This is especially true when these systems are series-linked manipulators that are flexible in any plane relative to each link as the errors are compounded.

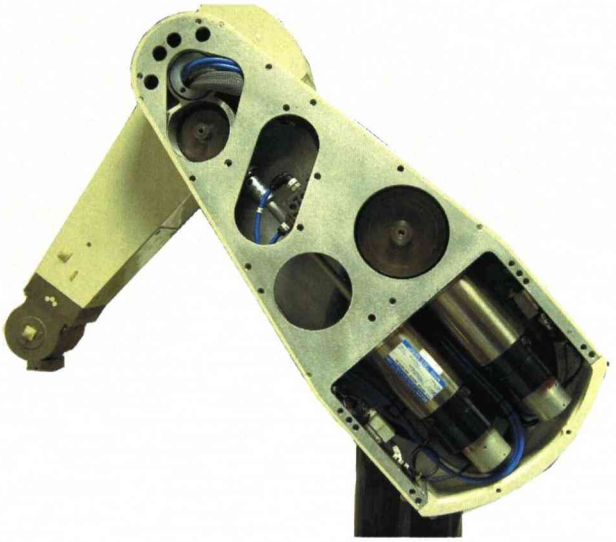

Figure 1.1: Image of the internal structure of a PUMA manipulator.

Figure 1.2: Image of the shoulder joint internal structure.

1.1 Motivation

Since the introduction of the field of robotics, they have been considered as the cutting edge of technology in the manufacturing and research areas. Inherent with cutting edge technology comes a high price and often high maintenance, where cost benefits are difficult to see without large production runs or high-cost parts. A large portion of the cost of robotic manipulators is determined by the size, complexity and rigidity of the link and drive structures.

As was explained, existing robot technology used in manufacturing and assembly is based on heavy, high precision links and joints. These are used to ensure the calculations of end effector position are correct, irrespective of forces involved in the processes the robot is used to control. By reducing the manipulator flex and backlash to a minimum, the robot can be assumed to be rigid and the control system simplified to use rigid models of the robots motion without end effector error. The cost of these high precision links and drives are great due very high tolerance manufacturing processes required to create them. Manufacturers need to ensure that they stay strictly within the required tolerances, as well as maintaining their resistance to wear.

With the ever decreasing cost-power ratio in computer and computer vision systems, it seems prudent to attempt to offset some of the cost of the drive and link system by utilising sensors effectively. Using the minimum number of sensors and maximising the information extracted from that sensor, the effectiveness of the control system can also be maximised.

This will be investigated on a manipulator which has had its rigidity deliberately degraded in order to simulate a system designed with lower tolerances, similar to that which could be expected if the manipulator were designed as a low-cost model. The system should use a high speed eye-in-hand camera system to identify unwanted link flex by direct tip-deflection measurement and counteract this without the aid of

other sensors, other than the manipulator's existing optical joint encoders. It was proposed that this could be used to negate the link compliance, using open-source software and off-the-shelf computer systems, with a future view to perfecting the control algorithms and compacting this onto an embedded PC board for further cost and space reduction.

Most existing industrial eye-in-hand camera controlled robots use the same rigid arms, and utilise the camera vision to calculate the end effector position and the position relative to the target. The visual feedback data is used, alongside models of the manipulator parameters, and multiple other sensor systems such as strain gauges and accelerometers to predict and measure the current position and re-position the manipulator appropriately with assumed knowledge of how the system will react . Due to the relatively slow feedback rate of the camera systems, these are generally a look-and-move system, with the model predicting how the manipulator will respond. These require complete and complex models of the manipulator to be generated, however unknown tolerances and payloads pose a problem to the error correction as the dynamic models become inaccurate.

These existing solutions are suitable to correct the small errors introduced by the high tolerance manipulators, however limited work has been done on correcting large errors introduced by more flexible links and less precise joint drives [3]. Visual servoing, the use of cameras to control end effector position, has been investigated quite heavily in recent years [4, 5, 6, 7, 8, 9, 10] including work on the simultaneous control and calibration of visual servoing systems and similar calibration techniques [11, 12, 13, 14]. The increasing capability of camera systems (higher frame rates and greater resolutions) and increasing affordable computational power open the way for more accurate position control and error correction.

The purpose of this work is to utilise the improving computer and vision hardware capabilities to investigate control over lower tolerance, higher compliance machines. That is, using high-speed visual control in an eye-in-hand configuration (EIH) to

allow the use of lower cost, less rigid manipulators. The availability of open-source, free, realtime operating systems also aids in this investigation and helps to demonstrate that the cost of manipulators can be reduced. This thesis investigates the feasibility of using a single end-effector sensor and high-speed control system to control a compliant robot system without extra sensor systems or extensive modeling.

1.1.1 Existing Work

The majority of applications for robots require the interaction of the tool of the manipulator with the environment or some item within the environment. In many situations, such as pick and place operations in known environments, the manipulator may only know the correct position of those parts within the world coordinate frame in order to interact as prescribed. This often means the process leading up to the robot station has to be controlled tightly and objects or tasks be placed accurately within that reference frame. Modern robots have high repeatability and good accuracy due to the high tolerance design of the manipulator and its control components. Within such an environment a task may be executed repeatedly without failure, however, should the object or manipulator contain some degree of uncertainty an open-loop manipulator control system is useless without a mechanical design which can accommodate such errors.

Such mechanical designs are prominent in subsea situations where the manipulator may be in motion or have been translated or rotated by the environment, explained by Snow [15], here the design of the tool forces accurate location of the item being grasped. Jongkind [16] also outlines an analytical framework for designing grippers for hazardous environments where fault-tolerance is required. These are suitable where workpiece locations are uncertain.

Similarly, more recent work by Mills [17] uses multi-robots (combining groups of

manipulators) with some structural compliance in the gripper in order to grasp a rigid object using force feedback and feedforward to prevent excessive forces being developed in the robot subsystems. All of these systems can be used to negate the problem of either manipulator or workpiece misalignment in a pick and place task, however in a non-contact situation where no gripper is used these solutions are not viable.

In situations such as robotic seam welding, the gripper is replaced with a permanent tool to facilitate the welding process, such as a tungsten tip and shield-gas feed nozzle; the manipulator may only come close to the workpiece. In such a situation, any manipulator uncertainty or misalignment of the workpiece can introduce significant errors and possibly damage the robot or workpiece. Here, there are two methods of dealing with the errors; the first being to minimise them by design of process and manipulator, the second being to actually measure the misalignment using some form of sensor system. Measurements can then be used to control the manipulator position to effectively remove the error by adjusting its location in the world coordinate frame on a per-workpiece basis.

One increasingly common method is to incorporate a camera system into the control loop, which is known as visual servoing. Visual servoing is the process of extracting information from an image in order to control a robotic system. It is a very broad area of research including areas such as high-speed image processing, kinematics and dynamics, real-time computing and control. Unlike each of these individual areas, visual servoing combines the results of these areas in order to produce a solution to the problem of real-time, closed loop feedback control of a manipulator. The first mention of the term "visual servo control" is thought to be Hill and Park [18] in 1979, although work on end effector position control using visual feedback was undertaken as early as 1973.

There are two primary methods of incorporating vision into the control loop. Eye-to-Hand or fixed camera systems are those where the camera hardware observes

the position of the target, environment and manipulator. Eye-in-Hand (EIH), or moving camera, systems are those where the camera is mounted to the end effector of the manipulator and observes the target and environment from a moving position. Hutchinson, Hager and Corke [19] detail how the field can be further broken down into position based visual servoing (PBVS) and image based visual servoing (IBVS). Position based servoing is done using a model of the camera, manipulator and target and corrects the pose based on these known models, whereas IBVS is done by directly measuring pose errors from the image itself in relation to the manipulator position. In PBVS the image is used to estimate the manipulator pose and error and a new pose is generated for the manipulator to correct for that pose error. In IBVS the error is considered purely in the image space and the goal is to reduce the error in that space, regardless of the actual pose.

Most systems employ a *dynamic look and move* architecture as this requires relatively low-frequency position feedback updates from the camera system to the control system. Dynamic look and move, as the name suggests, is performed by sequential scene observation, interpretation and then finally by motion of the joints. Because many robot controllers have features such as the PUMA robot's ALTER, Section 3.3.3, where the controller hardware accepts direct Cartesian adjustment inputs, this allows the visual servoing control to utilise existing idealised dynamic control solutions for the manipulator hardware. *Direct* visual servoing is a control structure where the visual servoing control system itself calculates the joint position and accelerations based on the parameters and values extracted from the image, this is considerably more complex due to the high level of nonlinearity in dynamics and creates a highly challenging control problem.

In EIH systems, where the camera cannot directly observe the end effector, an uncalibrated camera system cannot accurately position the manipulator [20]. That is to say, if the camera does not know its exact relationship to the end effector position, it cannot judge the position of the end effector unless it can see it. For

this reason, many EIH systems actually position the hand in front of the camera so that it is visible in relation to its target. With fixed camera vision systems, the manipulator position relative to the target is always measurable but they suffer from lens aberation near the extremities of the image and inaccuracies if the camera systems are disturbed from their calibrated positions. Stereo fixed camera systems offer some degree of insensitivity to camera translation and rotation after initial calibration [20, 21, 22].

Ultimately, by using an EIH camera observing the end effector and target, an estimation of their relative distance and speed of approach can be gained without models or previously known dimensions. In the presence of possible uncertainty of camera calibration relative to target or end effector, EIH systems with the tool visible in the image are the optimal solution as they provide direct measurement of end effector position. This comes at a cost as the visual control system has more detail to extract from the image which may increase computational power requirements, in turn reducing sampling rates. Calibration of the camera to the end effector position is usually done by either precise physical measurement and adjustment or by moving the robot to precise locations and assessing their position in the image created by the camera observing. Some systems [23, 24, 25] combine both fixed and moving cameras to gain additional accuracy and calibration ability.

Previous EIH research has relied almost completely on models providing the knowledge that the robot manipulator being controlled is rigid and responds as expected regardless of payload, without oscillation or varying model parameters. A detailed review of many of the approaches to, and principles of, visual servoing with rigid robots can be found in [26, 27]. These modeling principles have been applied more recently in [3, 28, 29, 30] who develop a complete model of the simple flexible manipulator system including stressing the importance of modeling the motors at the joints of the robot. It is interesting to note that, due to conflicting parameters between previous modeling work yielding similar results, discussed in [27, 31], Corke

[27] suggests the in-depth rigid body dynamics are relatively unimportant, at least in rigid manipulators.

Researchers in the area of force control have shown that as sensor bandwidth increases, it introduces instability in the control system as the control loop bandwidth is increased due to dynamic effects in the robot and sensor systems [32]. These problems are duplicated in visual control but are reported less due to the generally lower sampling rates quoted in literature. It could be assumed that this is due to limitations in the camera and computing technologies used in the research. Simulation results for rigid robot, high speed visual control [33] have suggested sampling rates of over 300Hz are required for stable and accurate control of a rigid three degree of freedom revolute robot.

Existing works make use of accurate models of the robotic manipulator, including estimations of errors due to flexibility, inertia in the links, payload and gravitational effects. They then attempt to control the manipulator as accurately as possible using these complex models and some degree of feedback from a multitude of sensors such as accelerometers and strain gauges [29, 34], often including adaptive control to "learn" about the parameters and operation of the manipulator while working [4, 9, 35] to improve performance.

In the this research, control of position was to be carried out using only the rigid-body kinematics, no model of errors, and a camera target observer to correct for errors in end effector pose. It was proposed that the high speed visual feedback of exact end effector error information is sufficient to remove the requirement for further modeling of the manipulator itself, or indeed further sensor systems. In order to assess this, the basic kinematic algorithms for the robot were overlaid with the data from the camera system and this data was used to calculate a corrected position for the arm in an iterative fashion without complex dynamic models. With the use of the high speed camera sensor, it is proposed that the system is capable of correcting plant or target misalignment, including vibration attenuation. With such

a system the dynamics of individual manipulators and payloads can be neglected or calculated online, leading to a vision system that can be applied to many poor-tolerance or flexible manipulators without complex modeling being required.

1.2 Solution Summary

It was proposed that the use of a high speed camera system can correct for the errors introduced by manipulator compliance and backlash. In order to assess this the experimental work was carried out on the PUMA 560 industrial robot. Though far from state of the art, the PUMA is still widely used in the academic and research environment as a robust, accurate and repeatable manipulator [36, 37, 38, 39, 40]. In this feasibility study, the manipulator was modified using a flexible link section added to the wrist joint in order to simulate a manipulator with lower rigidity. A PixeLINK A641 camera was mounted on the new manipulator end effector along with varying masses, simulating varying payloads, and rigid-body kinematics were used along with the high speed sensor system to show the possibility of using high speed visual error correction alone to negate the effects of manipulator compliance.

The control system was developed on a Linux operating system (OS) patched with the Real-Time Application Interface (RTAI), although the camera subsystem was developed on a Microsoft Windows^{TM} XP based system due to driver compatibility problems. The camera data was pre-processed before transmission, over dedicated LAN cabling, to the control computer. The manipulator communicated using the SLAVE interface software loaded into VAL, this was released by the manufacturer but rarely used in practice. Camera data rates of 350 frames per second were achieved, with the visual servoing control loops running at 1kHz. A summary of the implementation of each part of the system follows in the next section.

1.3 Implementation Summary

In order to allow the reader to understand the system as a whole without reading all chapters in detail, the following sections will give an overview of the experimental setup, the key software arrangement and a brief list of the chapter contents for further reading. The hardware used in the experimentation is composed of two PCs, a robotic manipulator (PUMA 560) with its own controller and finally a machine vision camera. The system can be broken down into three sections:

1. A camera subsystem composed of the machine vision camera, its host PC with processing software and a network connection to section 2.
2. A control subsystem composed of a high speed desktop PC with RTAI patched Linux (Fedora) OS, running separate trajectory and individual joint controllers.
3. The robot manipulator and its analogue controllers with accompanying interface computer running the SLAVE interface.

1.3.1 Camera Subsystem

The camera subsystem uses a PixeLINK A641 CMOS camera linked to a PC running the Microsoft Windows^{TM} XP operating system. With the ability to "window" on a small section of the image the A641 allows a 350 frame per second position tracking system. The host PC uses simplistic thresholding and parameter detection to determining the position of a target in respect to the centre of the windowed frame area. It then extracts the important information from the scene, sending this simplified data to the control subsystem via a dedicated 100BaseT ethernet link. This is described in both Chapter 4 and Section 5.1.

1.3.2 Control Subsystem

The main hub of control in the system is an Athlon XP based PC, running at 1.8GHz with 512Mb of RAM. This PC uses a version of Fedora, Redhat's open source Linux Distribution, patched with RTAI. RTAI is a high performance modification to the Linux system whereby Linux is run as a task within the RTAI system. This makes it preemptable by higher priority tasks and with high resolution timers and more effective scheduling converts the standard Linux system into a hard real-time system with extremely low jitter and latency. A full discussion of the requirements of the OS and the reasoning behind the selection of OS are detailed in Chapter 2.

Running in hard real-time, a trajectory controller system monitors the camera output and combines this with the kinematics of the manipulator in order to determine where the end effector is, where it should be in theory and how to correct any error present. This trajectory controller operates at 1kHz, combined with 350Hz camera feedback this is shown to be sufficient to counteract both steady state positional error and reduce vibrational effects with no other feedback.

The joint angles, both current and future, are also calculated by the control subsystem and transmitted back to the robot controller hardware by way of an RS-422 interface at 28ms intervals - a limiting factor in the ability to control vibration. More details of the control system are given in Chapter 5, the details of the robot kinematics and inverse kinematics are presented in Chapter 3.

The complete system was developed as a standalone controller, a modular software structure capable of controlling any hardware with some simple software interfacing. Future work to create and control a custom-made flexible robot and investigate futher advances are therefore possible.

1.3.3 Robot Hardware

The physical hardware being controlled, in this case was a Staubli PUMA 560 arm with six degrees of freedom, a standard industrial robot manipulator. A specially created flexible link was developed and designed to vibrate within a certain range of frequencies, with the camera mounted at its end effector. This is controlled by the standard Staubli motor control circuitry, however it uses the little-used SLAVE interface to communicate with the master computer - the control subsystem. The analogue servo controllers in the Staubli circuitry pose some restriction due to their own per-joint PID control, which has limitations. The robot hardware, its advantages and limitations, are explained in Chapter 3.

1.4 Thesis Outline

This thesis has been written in eight chapters, in order to fully explain the scope of the work covered. The remaining chapters take the following structure:

Chapter 2: Operating Systems

This chapter discusses the available computer operating systems, from the Microsoft Windows^{TM} range and standard Linux distributions to the real-time operating systems. A review of these systems and their key performance measures is given, the system selected for the research work is chosen and justified.

Chapter 3: Robot Manipulators

Discussed in this chapter is a review of early manipulator research and applications. The three primary types of industrial manipulators are presented, Cartesian, serial-link and parallel-link, including their advantages and disadvantages. The details of

the manipulator chosen for this work are given, including the kinematic equations linking world space positions to joint angles. This chapter also includes the design and testing of the flexible link used to purposefully degrade the robot's performance. A summery of the chapter is included.

Chapter 4: Vision Systems

Chapter 4 reviews the available computer vision hardware, including CMOS and CCD camera systems and their method of operation. The chapter includes a discussion of the key features of these and of the details of digitisating of images, such as sensor resolution, exposure time, speed and windowing features. Following this is specification of the camera used and the features exploited in this research. Details of the computer system used to host this camera are then given, the image processing and pattern recognition techniques used follow. Finally the camera's target hardware is described and the reasons for its construction are given.

Chapter 5: System Implementation

The System Implementation chapter explains how the technology in the previous three chapters was brought together. Initial sections describe the physical and software interfaces between the hardware and give an overview of the system as a whole, including the interface rates. Following this are details of the control software that was created to provide a proof of concept result, including initial configurations that were not as successful. Finally, a discussion of the two timescale nature of the motion of a compliant robotic manipulator, how the two timescale controller system overcomes this and possible alternative control system options is given.

Chapter 6: Results

This chapter begins with a discussion of the testing methods, then results from systematic tests of VAL open-loop control are given as a baseline comparison. Following this, single timescale control test results are presented including discussion of the implications of gain adjustments. Following this, two timescale tests are performed with the same systematic approach and the results explained. Further iterative gain adjustments are carried out to obtain significantly superior results to those of the previous tests.

Chapter 7: Discussion

A discussion of the achievements of the earlier chapters is given. Next, the advances and limitations of the control systems employed are outlined and the results are discussed in the context of their respective control system. The effect of the resonant frequency on the requirements of the control system are introduced.

Chapter 8: Conclusions and Future Work

In the final chapter the results and discussion are put into context and the overall contribution of the thesis is presented. The hardware design limitations that were found during the research are explained, including possible solutions. Likewise, software limitations are outlined including an assessment of the control techniques employed. A direction for future work is then proposed, both for the hardware implementation and control software.

Chapter 2

Operating Systems

2.1 Introduction

All complex computer systems require an operating system, an OS, in order to function. The computer system uses a Basic Input/Output System (BIOS) to identify and organise, on a very basic level, the hardware installed on the computer system. It has no real understanding of the details of the hardware or how to use it, it simply determines what resources each has and needs, and where to place them in the system. The OS takes the information collected by the BIOS and matches it to proprietary driver files in order to make use of all of the features of the hardware, and allow the user to interact with it. Originally the vast majority of computer systems ran on proprietary operating systems, such as the RC 4000 Multiprogramming system first used in 1969, due to differing hardware configurations and lack of standardisation.

To date there are hundreds of OS's that are specific to a certain piece of hardware. One of the more successful, however, was developed around the 1960s by AT&T Bell laboratories - Unix. As Unix developed in the specialist market the cost of computer systems fell and it was noted that computers could, in the future, be

used by the general public both at home and in offices, though this did not happen until later than initially expected. The first computers in general office use were the Commodore PET and BBC Micro, introduced in the late 1970s and early 1980s respectively. Over the course of the next decade and more, standardisation began to take hold. This meant more hardware was interchangeable and the competition reduced the price even further. One of the products of this process was Microsoft; Microsoft developed MS-DOS and Windows™ - command-line and graphical user interfaces respectively, which allowed the user to run and manage software on a generic hardware platform. Software could be run on different variations of similar hardware systems with the same results.

With the increase in use of computers in the home and office, due to their decreasing cost and increasing usability, the Microsoft brand took over 95 percent of the global desktop computer operating system market. In relatively recent years, systems other than IBM compatible PCs became powerful enough to have and use an operating system. This created a need for OS's tailored to their specific hardware implementation, yet adaptable enough to be used on different hardware platforms with the same (or similar) code. In addition, users who wish to do more than just write text documents or run one or two specialist windows-based applications (so called "power-users") found a need for more flexibility. Interchangeable packages based around a "kernel" OS were what was required.

In 1991 Linus Torvalds joined previous work on Unix-like operating system composed entirely of free software with his newly created "Linux" kernel. Over the following years the number of packages available to be run on the kernel grew until they rivalled the offerings of Microsoft. Differing sets of packages suited to different tasks - normal users, power users, developers, embedded systems - were created by the users themselves and re-combined in **distributions**. A combination of complex configuration, lack of user-friendliness and lack of commercial support meant the OS remained mainly in the academic and hobbyist user market. The basis

of this "Open Source" OS is the GNU license that it is distributed under - where anyone may take and use the software, and even modify it for their purposes, even commercial applications. The intention of this was that the users would, in turn, contribute by developing the software themselves and releasing the extra features and bug-fixes to the general public, and relying on the general public to feedback bug reports.

This has clearly been successful as Linux has secured a significant adoption within the web server market (despite Microsoft having developed its own Server editions of Windows^{TM}), as well as maintaining its use in the academic environment from desktop computers to supercomputers, embedded processors and other forms such as computer clusters. Different distributions are aimed at different targets and have package lists developed for those targets; some distributions are minimalist, designed to run on a small, low memory system such as an embedded processor within a network router. Others are designed to provide every piece of software the user could imagine - making a very powerful, yet free, desktop computer OS.

2.2 Performance

The operating systems of concern here are all multitasking environments, as opposed to sequential. Disregarding multiple processor systems, if a computer were to work exclusively on one software function until its completion, the user and the other software would sit suspended until the function had completed and released the processor to the next task. This is known as blocking. Although this usually gives the optimum performance for the active task, it means other functions, by definition, cannot be completed. Unimportant or low priority tasks can get full processor usage at the expense of critical tasks in this situation. Multitasking operating systems work around this problem by time-slicing. This is the process by which the OS takes a given amount of processor time and splits it into tiny fractions, typically around

10ms long. This is described in Figure 2.1 where a pair of tasks is executed in two different ways, one multitasked and one sequentially. In sequential operation, Task 2 must wait for Task 1 to be executed completely before it is given any processor time. In multitasking, Task 1 and Task 2 are interlaced, taking the same total time to execute but allowing both to run concurrently. A central *scheduler* keeps a register

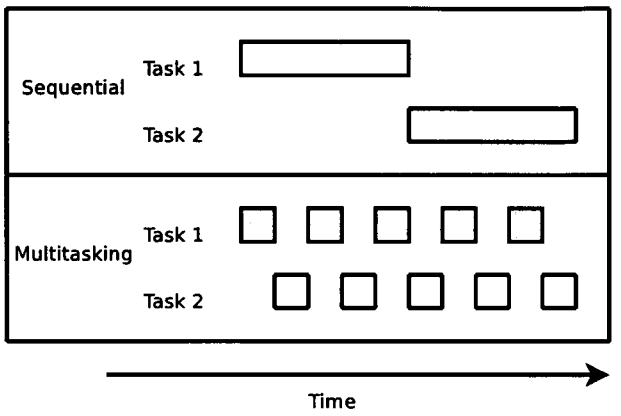

Figure 2.1: A diagrammatic view of multitasking in comparison with sequential execution.

of all running processes, their priority and the amount of Central Processing Unit (CPU) time they are using. During any given length of time, on average, several equal-priority tasks will get the same share of processor time. Process time will be interleaved with the other processes in a way that appears to the user to be running all processes at the same time - pseudoparallelism. This way a document can be typed and the spell-checker work "simultaneously". If the spell-checking portion of the word processor were to be performed without time-slicing, the user would have to initiate the spell checker and then wait until it had completed before they could continue typing. A higher priority task may take a larger share of the processor time, apparently slowing the other tasks in order to complete its task.

Jitter and Latency

This process of scheduled time-slicing itself takes time and processor power; the data in the CPU registers must be copied in and out for each process - this is called context switching. In order to provide low latency and low jitter, the variation in execution time of a periodic process, an OS must divide its CPU time into smaller fragments than normal. The problem generated by this is that the increased context switching rate itself takes up a larger percentage of the CPU time leaving less for the actual task.

In addition to running user and system processes, the system must respond to immediate calls for action which are given a much higher priority than a normal task. These are generally hard-wired to the CPU of the system, or set up via a Programmable Interrupt Controller - a PIC . In a Windows^{TM} type of environment the handling of interrupts is indistinguishable in normal use. The mouse sends interrupts to the OS in order to inform it of its position relative to the last position, yet the user doesn't see a degradation in performance of the overall system when they move the mouse.

In order to assess the performance of a given operating system, for the given task of hardware control, we wish to determine the jitter and latency. Assuming a process is set to run with a fixed period, P, in the order of one second to update the output variables from a process to screen, but the process is called just as another process begins its 10ms slice. In a normal OS the scheduler resolution, around 10ms, would not appear to affect the accuracy of the update timing; P + 10ms results in a 1% delay. If one then required another process to be executed at 20ms intervals, perhaps to control a motor position, the scheduler resolution becomes highly relevant and can detrimentally effect the process; P + 10ms results in a 50% error in timing. In a real-hardware control, sample times may be on a scale of μs, a 10ms scheduling would make this process impossible. To add to this, even if the 10ms resolution

could be guaranteed and the process had highest priority, the period of execution would always differ from 10ms due to varying context switching overheads - this is called jitter and is a measurement of variation from the prescribed execution period, high speed hardware control requires low jitter.

The second important factor is latency. Latency is the time between hardware raising an interrupt flag, or scheduling the start of a process, and the actual execution of the interrupt handler routine or process. As we will investigate later, real-time operating systems aim to minimise this to ensure the process of collecting information from the hardware is executed when it is expected.

Initially, in this research, the investigation of the operating system began with an assessment of the ability to access the general-purpose serial communications port of the computer, COM1. This was required to communicate with the manipulator controller hardware at a rate of 19,200bps and at a defined time. The manipulators SLAVE interface software transmits the current manipulator status, including encoded joint angles and faults in communication and hardware approximately every 28ms. It requires a timely acknowledgement of this transmission, expecting a response to be sent to it almost simultaneously - with a very narrow timing margin. If it does not receive a character within approximately 4ms it assumes the communications have failed and aborts the communications link attempt.

In addition to the requirements of SLAVE communications, the overall operation and control of the manipulator requires a reliable, regular execution of control calculations. These requirements mean the OS must be capable of low-jitter, low latency scheduling. In essence, the periodic calculations need to be performed at the time specified and interrupt handling must be executed as soon as possible after the interrupt is signalled - the interface with the human user is of secondary importance.

As mentioned earlier, the problems associated with low latency, low jitter operation are based around the fact that in order to reduce latency the processor and

hardware has to spend more time context switching (context switch overhead) and less time executing.

2.3 Real-time

Now that we have defined how general operating systems handle a set of processes we need to consider the requirements of the task of robot control with visual servoing. The process of controlling a manipulator requires rapid and repeated measurements of, and feedback to, the robot hardware.

The phrase **real-time** is a loosely defined term, with differing specification depending on the application within which it is being used and the marketing used to sell it. There are, however, three essentially different sub-definitions that describe all. These are commonly known as Human real-time, Soft real-time and Hard real-time. These are determined by the requirements of the system to be controlled.

2.3.1 Human Real-time

Human real-time can be determined to be a process or data update that occurs frequently enough for the human to maintain control of it, or otherwise observe the results of, to a reasonable extent. The definition of "reasonable" depends on the process at hand; For example, a human monitoring and controlling the temperature of a swimming pool could consider a digital thermometer output with a refresh rate of fifteen minutes to be real-time, considering the rate at which the large volume of water would change temperature.

2.3.2 Soft Real-time

Soft real-time is simply an extension of the specification for Human real-time; a process or update rate that occurs rapidly enough that any loss or jitter in the

process does not cause significant degradation to the process being controlled or observed. An excellent example of this is given in [41], a video recording or display - at 25 fps the viewer can enjoy the overall process and will be largely unaware of slight mistakes in the image or even missing frames.

2.3.3 Hard Real-time

Hard real-time is the tightest, and some would consider true, definition of real-time. A hard real-time process relies on **guaranteed** and **repeatable** responses with minimal jitter. These could be software generated responses or interrupt-generated responses. As the response cannot be delayed or preempted, the system cannot hide slow responses behind fast averages and for this reason hard real-time systems are used to control critical and rapid processes.

For the purposes of this work the general target is to reduce the cost of robotic manipulators and their control systems. This, in turn, disadvantages many of the proprietary operating systems - licences for their use and the limited scope of usable hardware requires the development of yet more proprietary hardware and software - compounding the problem. Therefore, in order to remain within this aim, only cheaply available and easily programmable, reconfigurable operating systems were investigated. These were Microsoft Windows™ 2000, Microsoft Windows™ XP, Microsoft Windows™ CE, Linux (several possible distributions), and a real-time patched Linux. Although the cost of the Windows™ operating systems is high when purchased seperately, businesses wishing to create a number of systems around the OS can purchase the right to distribute their hardware with greatly reduced license costs.

2.4 Testing

Initial testing of the operating systems was performed by assessing the response to the serial SLAVE interface communications. As mentioned earlier, this was required to communicate via the serial port in packets of data. However it requires a timely acknowledgement of this transmission, expecting a response to be sent to it within a very narrow timing margin. If it does not receive a character within approximately 4ms it assumes the communications have failed and aborts the communications link attempt.

Simple code was used to reply to the SLAVE software via the RS-232 line, with exactly the same packet of information as was sent from SLAVE. This is a standard technique suggested by the manufacturer in order to ensure communications timing and packet construction is correct before actually moving the manipulator. Incorrect data packets can create **extremely** large joint accelerations and random manipulator movements that could damage the joint motors or gearing. Also, the structure of the robot itself could be damaged should it contact a solid obstacle within its envelope of movement.

2.4.1 Windows™

The Microsoft windows series has evolved over time from an application, itself hosted by the MS-DOS operating system, to a fully developed OS. During this time it has gone through several updates that have made step changes to the way the software handles the hardware and what access is granted to the users and their software. This has been good for security of the OS and ease of use by the general public. However, this has in turn made direct hardware access harder to achieve for the software developer.

In early versions of Windows™ (95, 98 and 2000) the developer could have direct access to the ports, their registers and FIFOs . This was removed in later

editions in order to standardise access methods and prevent malicious hardware access. This, in turn, means that the developer is effectively asking the OS's driver to perform a task and relying on it to do so in a timely and appropriate manner. There are modifications (third party drivers) that can be used to call direct access to the ports but these are relatively poorly documented.

Windows™ 2000 was used, initially as a learning process, to create the code to echo the data back to the SLAVE interface in C++. The code is extremely simple in order to assess the best possible turnaround time for the packets of data. A test system was set up and a breakout board was created at the robot controller serial line drivers in order to monitor and measure the communications in each direction with a dual channel oscilloscope. After many attempts the SLAVE controller would not establish a link to the Windows™ 2000 PC; upon inspection of the signal this appears to be due to a 7.6ms delay between reception of the first byte and transmission of the first return byte. In order to confirm that this delay was the cause of the communications error a simple Field Programmable Gate Array (FPGA) processor was constructed in order to return *any* byte immediately. This code functioned correctly and the SLAVE software continued until the packet error-checking stage, at which time it aborted, reporting a protocol error. According to the timing diagrams in the SLAVE interface specification, 4ms is the shortest timeframe in which events occur - from this it was deduced that the response from the external control computer must begin within 4ms in order for the SLAVE interface to accept the communication.

Despite several attempts to shorten the response time of the Windows 2000 machine, the 7.6ms delay could not be reduced; the code was re-written for assessment on a Windows™ XP machine. Here too the delay existed, although slightly shorter at 6.9ms. Windows™ was never designed as a real-time operating system, as described in [42], although XP can be seen to be suitable for soft realtime operation. However, it makes no guarantees as to upper and lower bounds for latency and

jitter which makes it unsuitable for a task requiring hard real-time control. The work of [42] was not available during the period in which the OS was being selected, however, after a review of literature such as [43] it was decided that further testing with desktop Windows™ was futile; all literature agreed that Windows™ was only capable of soft real-time control and even then only under strict conditions that were often not fully described.

The use of Windows™, as a multi-tasking OS, was therefore ruled out. The remaining options were the real-time operating system "Windows™ CE", Linux, or a RT (real-time) version of Linux. Windows CE was avoided due to the licence costs and limited freely available development support. Standard Linux was not tested as the process of installing and configuring for a real-time Linux system was a matter of a simple patch and compile of normal Linux - the extra effort would ensure the operation of a true, free, Real-Time Operating System (RTOS).

2.4.2 Real-time Linux

There are two methods of making a "Real-time" version of linux, as described by Bird [44]. These are:

- Fine tune the way the kernel time-slices the tasks and prioritises them for execution.
- Create a separate real-time kernel and run the normal Linux kernel as a separate task within the new kernel.

Although these are both suitable options it is the view of the Author that the second option is executed more elegantly and with greater support by a larger cross section of developers as improved preemption solutions are many and diverse. The guarantees required by high speed robotic control cannot be given by improved kernel preemption, not least due to the way the kernels prioritise task execution and still

ultimately queue the RT task with non-RT tasks. Therefore the use of a separate kernel, running Linux as a preemptible process (unless otherwise instructed) was chosen for this research.

In order to discuss this further it is important to introduce the kernel space and user space concepts. Kernel space exists within Linux as the core operating system area. It has its own memory address range separate from the user space applications to facilitate a robust system - if applications cannot access the same memory areas they cannot corrupt the kernel operations.

2.4.3 RTAI vs. RT-Linux

Within the Linux community, there are several patches and modifications to make the basic kernel an RT version. The two most popular are RT-Linux (Real-Time Linux) and RTAI (Real-Time Application Interface). All programming systems and libraries of code rely on an API, this is the structure and functions by which other source code can execute items from that system. The RT-Linux API and structure was present before RTAI, however a branch in development occurred over disagreements over standards and development continued separately creating RTAI as an alternative. RT-Linux is still available but does not seem to enjoy active development any longer (as of 2001) and it seems to have fallen behind RTAI in terms of the number of supported microprocessors.

RTAI, however, has emerged and is still under constant review and development. The current version at time of writing is 3.5 and is provided with patches for both v2.6 and v2.4 Linux kernels, covering most systems. In addition, [45] provides a detailed test of both RTAI and RT-Linux under stressed and unstressed conditions and finds RTAI holds the significant advantage - the standard deviation of jitter on a 2kHz task with a loaded computer was over 50 times larger on RT-Linux as on RTAI. Aarno, [45], carried out tests on a 400MHz system; tests in this research were

carried out on a 1.8GHz machine. As Aarno's results were sufficient for the purpose of this thesis, no testing was carried out to prove the 1.8GHz machine was more capable.

2.4.4 RTAI - Key Features

RTAI has the ability to create **shared memory**, between RTAI and Linux, which allows processes to share data. Due to the fact that the kernel space memory needs to remain protected from mistakes or malicious damage by software run in user space, normal processes interact with the kernel through FIFO's and mailboxes. RTAI, however provides a method of allowing the same area of memory to be addressed by both sides of the kernel/user space barrier. This is useful if many processes use this data as the solution using mailboxes or FIFO implementation would rapidly become complex with increasing numbers of processes.

RTAI provides mailboxes with messages ordered in first-in-first-out order, much like a FIFO. Different sizes of messages are allowed, as well as multiple senders and receivers, which can read and write messages to the same mailbox. There are several sending and receiving functions that provide a lot of flexibility. Blocking operations are those which force the task execution to halt until the operation returns an answer, while non-blocking functions return immediately with information as to whether the operation was successful. Timed message sends allow time limits to be set on message transfer, with timeout preventing a process waiting for too long before returning if it is blocked for a period of time. Partial message sending allows over-sized messages to be transmitted in blocks if the data cannot be sent in one go.

RTAI also provides several synchronisation utilities including mutexes, conditional variables and semaphores. Mutexes, short for mutually exclusive, are locks on certain pairs of sections of code which prevent the execution of the other section if one is within that critical section. This prevents data being corrupted by two processes

working on the same variable. Semaphores, the other utility used extensively in this research, are flags around a certain variable or section of code, similar in operation to a mutex. When a section of code protected by a semaphore is executed, the function *rt_sem_wait*() is called and the flag is tested for its value. If the value is above a set level the semaphore is "taken" by the task and the value of the semaphore is reduced. Other sections of code, using the same variables as the first section of code, test the semaphore when they come to use those variables, again the function *rt_sem_wait*() is called and this time the flag is found to be taken. The second task then waits on the semaphore flag in a queue for execution. When the original section has finished its use of that area it executes the *rt_sem_signal*() function to increase the semaphore value and allow the next task in the queue to have access. These can be used to control the order in which processes access data and, in turn, the order in which those tasks execute.

LXRT

RTAI has a well developed and tested user-space real-time system called LXRT. Except for certain key items it shares the same API as the kernel-space realtime tasks, allowing easy transition between programming of the two. There are three different implementations:

- LXRT services, "soft-hard" real-time in user space, with user-space API. LXRT loads as a module that allows the programmer to use RTAI services in user space. This is implemented by using a RTAI (kernel) process for every LXRT process - a task *buddy* that executes the actual real-time functions. This makes it possible to use any Linux system calls from a LXRT process, as well as RTAI services. LXRT tasks can communicate with kernel space tasks by using the same API, share memory, mailboxes etc.

- Extended LXRT, or "hard-hard" real-time in user space. This is an extension of the original LXRT facility. By calling the function *rt_make_hard_real_time*() any LXRT process is given an even higher priority in the system. These processes do not execute any Linux system call, as that can lead to a task switch, taking the system out of hard real-time - this also risks a deadlock situation.

- Recently a "Mini LXRT" tasklet facility has been provided. These are essentially a way to initiate RTAI functions from within the user space.

2.5 Summary

In this chapter a review of both Linux and Windows^{TM} operating systems, including real-time operating systems, was presented. A review of the common terms used in the field, such as soft and hard real-time, was also presented and a discussion of the critical performance indicators of real-time systems was given. General Windows systems were avoided due to their time sharing nature which provides no guarantees on task execution time. Real-time Linux patches, including improved schedulers and whole kernel replacements, were investigated and the differences outlined.

Previous literature on the subject was consulted and testing of a number of the operating systems provided the basis for the selection of the RTAI real-time Linux patch, developed by DIAPM [46]. The critical details of this patch and the differences between it and its direct competition were given and some of the important features of the selected operating system were discussed.

Chapter 3

Robot Manipulators

Robot manipulators have been in use in industry since 1961, when the "Unimate" was introduced on an automotive manufacturing line to remove hot workpieces from a casting process. Weighing 1800kg, the Unimate was extremely useful despite its limited manipulation capabilities, and provided basic welding and pick-and-place operations alongside humans who were glad to hand over dangerous, hard labour to a programmable robot. Unimate continued to make robots of differing capabilities and sizes, though the most widely known are the PUMA series robots which consist of a serial-link arm with either five or six degrees of freedom and an end effector at the tip.

There are other styles of manipulator that are in widespread use, some of these are detailed in the next section. Although the term "robot" commonly invokes images of either humanoid beings or serial-link robots, many designs exist and are each suited to certain tasks. Along with such thoughts, often arising is the theme of artificial intelligence (AI) - an area that, despite in-depth research work across the world, remains absent from industrial robots in the form commonly associated with the term. The field of AI actually includes some very basic tools such as simple reasoning and some complex and useful areas such as neural networks. The term

robot can be applied to very simple machines; The prerequisites for categorisation are as simple as being a machine that is able to sense and interact with its environment, and that is programmable. As the field advances the number of these conditions is continuously being increased.

Robots are often categorised by the number of degrees of freedom they have. Degrees of freedom are a set of independent displacements that can specify the position and orientation of a body. A rigid body can have up to six degrees of freedom, specified as translational (x,y,z in Cartesian coordinates) and rotational (often classed as roll, pitch and yaw). With these six degrees of freedom, any orientation in 3D space can be specified. With more than six degrees of freedom, robots can not only move to any position and end effector angle, but do so with more than one solution. This allows the robot to perform object avoidance, giving multiple ways of shaping the arm to avoid the environment or other robots in its original path.

3.1 Manipulator Types

Cartesian

Cartesian manipulators are so named due to their position and motion being controlled by a beam system, in which each beam lies in an axis of the Cartesian coordinate system, Figure 3.1. Motion of a "head" along each beam allows translation in that axis, and with three axes of translation the tool can be positioned in any place in the robot's workspace by simultaneous or separate movement. These are often referred to as gantry robots due to the fact that they can be scaled up to extremely large sizes in order to deal with very heavy loads and work over very large operating envelopes. In their standard form they are three DoF robots, but they can be fitted with a separate spherical wrist joint in order to allow pose control for angled approach and manipulation. They are often used for tasks such as pick

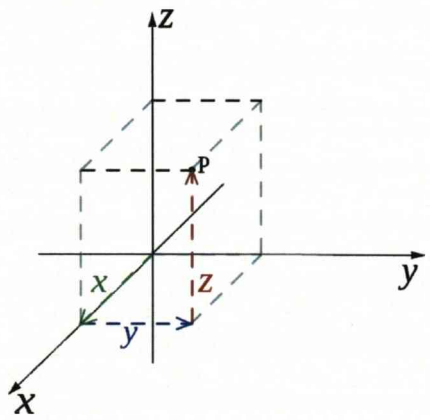

Figure 3.1: The Cartesian coordinate system.

and place operations, soldering and screw fastening, as well as machine cutting and welding on flat sheets as the design lends itself particularly well to these tasks. A major disadvantage to this type of robot is that if it is used over large operating envelopes, the beams of the system need to be extremely rigid in order for the robot to maintain its accuracy. This is compounded by the fact that the workpiece cannot be larger than the operating envelope of the robot as it would obstruct the motion of the beams.

Serial Link

Robots with a serial link structure are named due to the end-to-end nature of their construction. They are comprised of a series of interconnected links, each attached to the former by either a revolute or translational joint, and tend to assume the general shape and motion of a biological limb, Figure 3.2. These are available in many configurations and are widely adaptable to almost any task within a small working envelope, although large-envelope versions are available they are in the minority - the space shuttle loading bay arm is one such exception, with a length of over 15m and a full six degrees of freedom. Longer arms tend to be manufactured from composite materials in order to maintain accuracy by reducing weight and increasing stiffness.

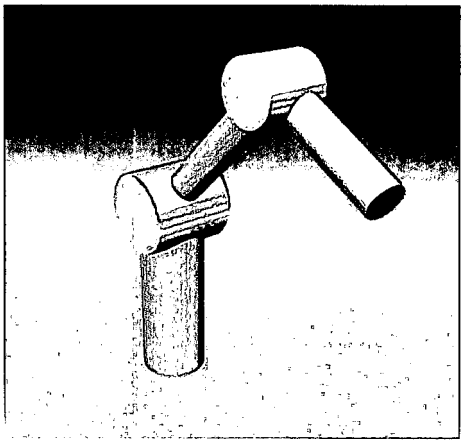

Figure 3.2: A serial link manipulator structure

Serial link robots are the classical industrial manipulator, models with a low number of degrees of freedom are often used for simple tasks such as planar motion pick-and-place operations. Tasks that require complex positioning and trajectory control, such as spot-welding inside an automotive body panel, will use a six or seven degree of freedom robot. These are capable of approaching any position in their primary working envelope from almost any angle. They are generally limited to a small envelope due to constraints such as the extremely large torques required to suspend a payload at large distances from the base. Serial link robots are available in both servomotor and pneumatic actuation styles and can be tailored to a wide array of payloads and tasks, from micro manipulation tasks accurate to tiny tolerances to high speed lifting and packing of heavy payloads. The open chain structure, however, also suffers from susceptibility to large end effector errors due to link flex and joint tolerance; investigation into the removal of these errors is the objective of this thesis.

The extreme of the open chain serial manipulator is the snake-arm or serpentine manipulator, such as that developed at JPL [47] and others [48, 49]. Manipulators with more than six degrees of freedom are considered to have redundant degrees of freedom, snake-arm robots are often considered to be "hyper-redundant" manipulators - having vastly more degrees of freedom than are necessary to reach any position

in 3D space. They are commonly used to manoeuvre around inside enclosed spaces where object avoidance is key.

Parallel Link

Many combinations of manipulator structure are possible, in industry there is one other main category of robot that, despite an almost equal development time, has not enjoyed the same prolific use. Parallel link manipulators are constructed with two or more (usually at least three) parallel links connected from the base to the tool. Due to their parallel link design they are considerably more rigid than their serial counterparts, and can be used to move very heavy loads due to the load being shared across many actuators. In addition to the increased rigidity, any errors due to flex are averaged, not compounded as with serial linked robots. This allows robots such as the Fanuc F-200iB, Figure 3.3, to manipulate 100kg payloads with the same repeatability as is possible on the PUMA 560 with only a 2.5kg payload. With

Figure 3.3: A Fanuc F-200iB parallel manipulator.

the control actuators generally mounted at the fixed end of the links, the links do not have to overcome the inertia of the other links drive gear. This gives another

advantage in the form of speed, the Adept Quattro recently claiming 250 vision guided, random pick and place operations per minute. The characteristic flaw with parallel robots is, outside of a very small "optimal" operating envelope, the reach affects the number of degrees of freedom, limiting the tool pose capabilities.

3.2 Requirements

In order to assess the performance of the proposed system, any reasonably sized accurate and repeatable manipulator is considered suitable, although the simpler and more rigid the test system is, the better the results can be compared. The key features of the testbed manipulator are:

- Rigidity
- Zero Backlash
- Ease of modification
- Powerful joint actuators

While the purpose of the experimentation is to show that compliance is not a problem with modern sensor systems, a fully rigid, accurate and repeatable manipulator allows degradation of the rigidity and accuracy in a systematic manner. This manner can be prescribed to suit the experimentation and the results can, therefore, be trusted to be due only to predetermined error and not some error introduced by an unknown source.

The manipulator available and considered capable of this research was a PUMA 560, a six degree of freedom manipulator capable of manipulating a 2.5kg payload. The PUMA, although no longer state of the art in industrial manipulators, is widely used in academic settings to prove concepts as well as being a dependable manipulator in industry and is often used as a spot welding machine. The original

price of such manipulators was very high and parts remain extremely expensive. It is capable of a maximum accurate-path speed of $1ms^{-1}$ and has positional repeatability of 0.2mm. The manipulator has a positional control frequency of just less than 36Hz when accessed through its SLAVE interface system, an interface providing direct access to both the joint angle encoder information and the target joint angle values. A custom crafted manipulator was considered but was not attempted due to time and machining constraints within the department.

3.3 PUMA 560 Manipulator

The basis for all of the experimentation was a UNIMATE PUMA 560, six DoF, manipulator. The robot consists of two main parts; the manipulator arm itself and the control computer. These are separated by 5m of armoured cabling and the arm section mounted on a rigid steel structure.

3.3.1 Arm

The manipulator arm is equipped with six revolute joints, each actuated by their own DC servomotor though a gearing and drive system. Depending on the joint, each motor has differing gearing but each is fitted with a rotary joint encoder. Each of the revolute joints has different maximum operating angles, which are determined by the physical design of the arm. The encoders provide the control hardware with joint position values of -180 degrees to +180 degrees with 16-bit resolution, although not all of the range is usable as noted in Figure 3.4.

As can be seen in Figure 3.4, the manipulator is a serial-link arm, comprised of five primary sections, where each of the first three revolute joints is connected to the last at its distal end, the final three links form a spherical joint - the sections are enumerated below.

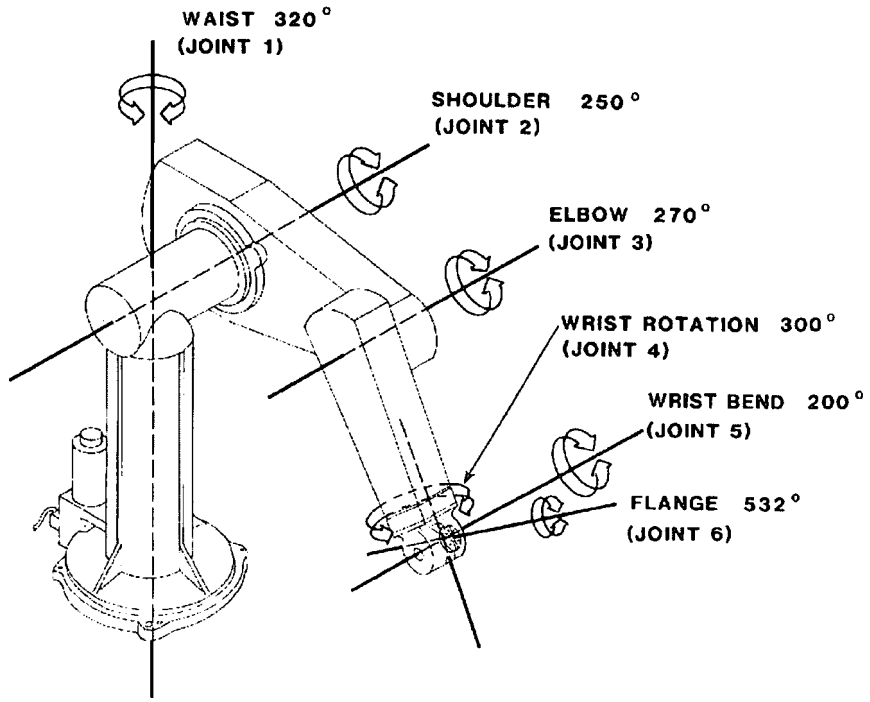

Figure 3.4: PUMA 560 joint orientations and maximum operating angles.

1. **Trunk**, aligned with the rotation of Joint 1, supporting the arm.
2. **Shoulder**, providing the axis of rotation for Joint 2, housing Joint 2's bearings.
3. **Upper arm**, containing the drive gears and motors to control Joints 2 and 3.
4. **Forearm**, containing the drive gears and motors to control Joints 4, 5 and 6.
5. **Wrist**, a spherical joint containing Joints 4, 5 and 6 with coincidental axes of rotation, allowing pose control.

It can be shown that, from the maximum resolution and angular displacements, the joint encoder resolution is given by:

$$Resolution = \frac{360}{65536} = 5.49 \times 10^{-3}[degrees] \tag{3.1}$$

This gives an angular resolution of almost 0.005 degrees allowing very precise joint position control by the on-board controller. This is not the case for the Joint 6, which can rotate through more than 360 degrees and therefore the 16-bit resolution is extended across a 720 degree range, halving the angular resolution on that joint, as demonstrated in Table 3.1.

Table 3.1: Angular resolutions of the six revolute joints in a PUMA 560.

Joint	*Maximum Range(Deg.)*	*Angular Resolution(Deg.)*
1	320	0.005
2	250	0.005
3	270	0.005
4	300	0.005
5	200	0.005
6	532	0.010

Using the first three links the robotic arm can manipulate the tool into a desired space in its spherical working environment. The fourth, fifth and sixth joints consequently can be used to rotate the tool to any given pose, assuming it is within the allowable joint range.

3.3.2 Controller

Under normal circumstances the manipulator arm is controlled by the main control unit which has the VAL II robot control language installed, referred to simply as VAL . The version of VAL is not significant in this investigation as the control system used completely replaces the VAL system. In normal usage the manipulator would be programmed on the controller using VAL, this is a basic command system that allows the user to "record" a series of positions and movements into a file. This

set of instructions may then be looped indefinitely and even interfaced with basic external sensor systems to allow the robot to act and react within a manufacturing environment. VAL takes the kinematics and dynamics of the robot and computes the best way in which to move the joints to the required next position. Positions can be programmed using numeric entry of the position and pose, or by the use of the teach pendant to position the arm as required before recording that position in memory. When positions are entered by the teach pendant, the repeatability of the PUMA allows that position to be returned to within ± 0.1mm. If that position had been programmed offline in a VAL programme, the actual position reached by the arm would be less predictable, due to the fact that the high-resolution encoders need re-calibrating at every power-up.

The control system is based on an LSI/11 computer system which computes the kinematics and dynamics of the joint motions required to actuate the end effector movements specified. The architecture of the main robot control computer can be seen in Figure 3.5. An in depth explanation of controller arrangement can be found in [27]. As can be seen in the schematic, the main LSI/11 computes the required joint trajectory relative to the current values and passes that information through the parallel interface board. This routes the information to the digital servo drivers by way of a six channel, bidirectional bus. In the Mk-III controller at the University, the digital and analogue servo boards are combined into one board but are shown here for clarity. The digital section of the servo driver board reads the current encoder positions relative to the latest instructed position and executes a position control loop. It also deals with the setting of basic parameters for each joint. The digital boards are then connected to the analogue boards which carry out velocity and current control, which in turn pass an analogue signal to the amplifiers that actually drive the joint motors. The digital servo control loops run on a clock frequency of 1MHz and the position control is updated at a rate of 1kHz.

The LSI/11 is provided with the VAL programming language, VAL is a control

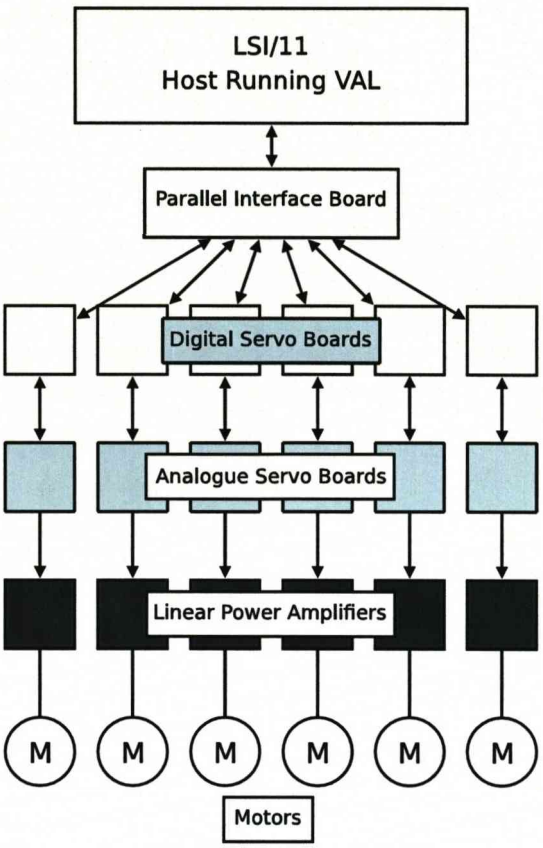

Figure 3.5: PUMA Controller architecture.

system and language designed specifically for use with Unimation industrial robots, such as the PUMA series. The language is stored in the systems non-volatile memory and is loaded at each power cycle. It includes the routines and libraries used to control the servo positions, including the kinematics and dynamics of the arm being controlled. The language interprets text commands in the form of direct keyboard input or by interpretation from a text-based file known as a program. VAL features continuous trajectory computation and allows complex motions to be instructed with a simplified command structure, making the robot easily programmable by a relatively un-skilled operator. Under normal operation, each line of the program is interpreted in order and the robot moves in the prescribed manner between locations and orientations. It assumes it is fully rigid and is independent of its environment or payload.

The speed of operation can be increased or decreased using the SPEED command, investigated further in Chapter 6, which scales the maximum tool translational speed (not the joint speed specifically) from fractional values to multiple hundreds. The speed figures are not absolute measurements, but a low setting can protect the joints from excessive accelerations with heavy loads or around singularities. They do not directly affect the operation of the robot via the teach pendant, which is used to manouvre the robot manually. Positional accuracy and repeatability is only quoted up to speed 100, however payload mass can affect this, which is the reason for general recommendations that a slow speed is used with larger masses.

3.3.3 External Alter and SLAVE

As well as the fully programmed motion created by a VAL program, the controller and VAL also provide an "Alter" feature. This allows corrections to the robots current position or trajectory by low bandwidth commands over a serial connection. These commands are transmitted at a refresh rate of nearly 36Hz (28ms period). This is often used in situations such as force control on assembly lines. This could be used for visual feedback although, despite the age and common use of the hardware, there is very limited information about the actual requirements of the alter command requirements available. Also available, but little used and unsupported, is the SLAVE interface. The SLAVE interface totally bypasses the VAL system, including its kinematics and dynamics, and implements a low level bidirectional joint position control as can be seen in Chapter 5.1.

The interface operates at an update period of 28ms, over a 19,200 baud serial RS-422 line. In each 28ms interval the current position and state of the joints and gripper is transmitted to the new host computer that is taking over from the VAL system. In return the SLAVE system requires immediate transmission of new joint positions in order to remain active, even if these positions are the same as the current

position. As shown in Figure 3.6, in effect the SLAVE system simply interfaces an external controller computer with the digital servo boards whilst providing some level of hardware error detection. By allowing access to raw joint locations, the SLAVE system allows the programmer the freedom to create and test their own control methods, however it does come at a price. The only protection from over actuating or over accelerating the joints are the limit stops and the current limiters in the main servo amplifiers. This has potential for mechanical and human damage, from a system which is not bug-free. The risks involved were considered to be lower than the potential benefit from a system allowing total access to apparently unrestricted per-joint positional control.

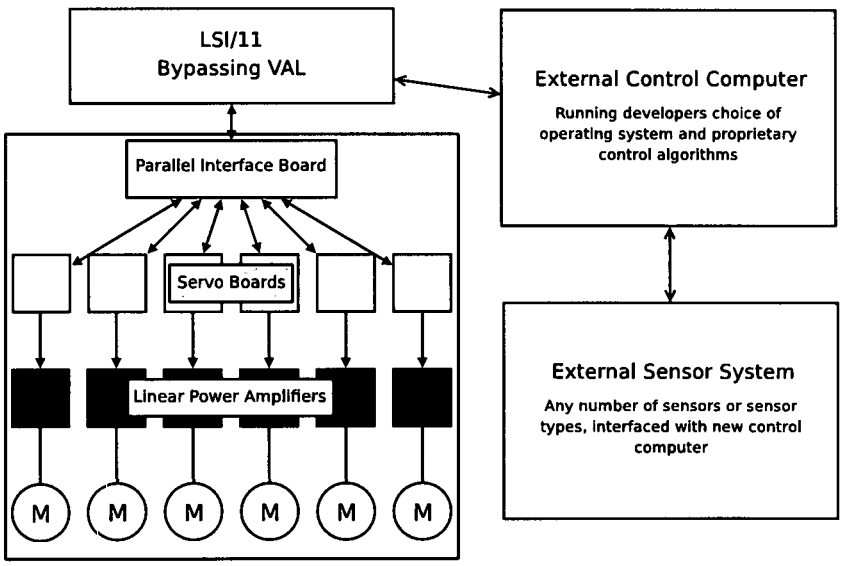

Figure 3.6: SLAVE's VAL Bypass architecture.

3.4 Flexible Link Design

In order to produce uncertainty in the arm position, simulating a low cost or lightweight flexible manipulator, an extension to the PUMA manipulator was created. In order to limit the arm compliance to a single plane of action, the extension

was designed in a manner that restricted compliance in other directions. Figure 3.7 is a diagram of the compliant link used in testing, showing the important dimensions of the beam, which was constructed from mild steel in order to provide good elasticity and a large end effector displacement before plastic yielding is reached; the details of the design are considered here. Of importance to the experimentation, the natural

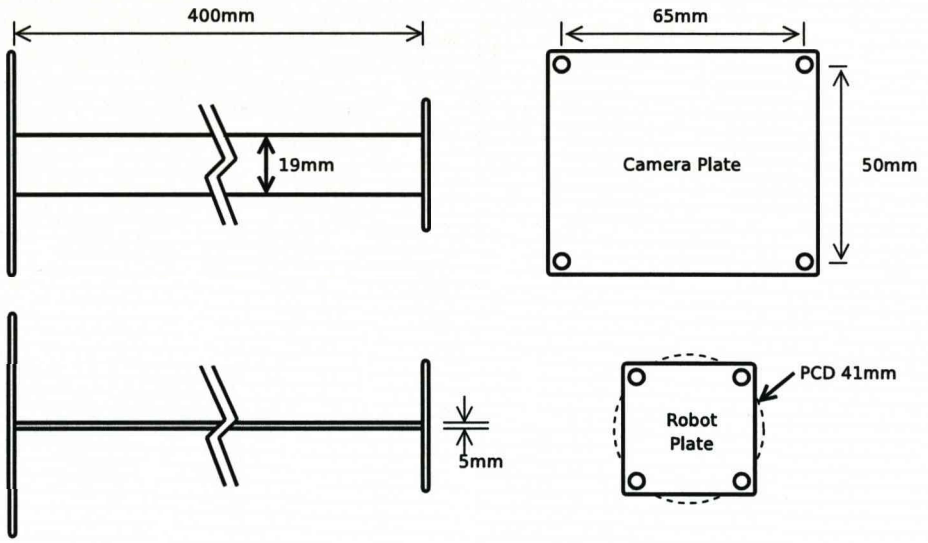

Figure 3.7: Key features of the flexible link attached as tool.

frequency ω_n, of the beam needed to be within a range that could be considered likely in an application of the overall proposed system. The suggested application was robotic welding - a line following task where, dependant on gravitational effects and the mass of the welding head selected, the end effector loading and oscillation could be considered variable and unknown. Large displacement, high frequency oscillations are unlikely in such a line-following and moderate pace operation where step changes in target position are generally less than 10mm and occur rarely due to the nature of the task. Also in consideration was the update rate of SLAVE; the task of controlling the compliance could lie outside the physical capabilities of the robot if the natural frequency is too close to the update rate of the arm. Chapter 5 explains why a natural frequency of 0 to 10Hz was considered within the robot capabilities and the beam designed with this in mind, with the ability to modify the

natural frequency by adding or subtracting mass from the end effector to provide a range of experimental data.

In order to calculate the size of beam required, the fundamental beam bending equations must be known and some simple assumptions made about the material and the manner in which the beam would be bending. Considering the case of manipulator droop, as proposed in the application above, it can be assumed that the beam will bend as a cantilever. Simple beam bending can be described by equation 3.2 from [50], where M is the applied bending moment, I is the second moment of area of the beam being bent, σ is the stress due to bending at a distance y from the neutral axis, E is the materials Young's modulus and finally R is the radius of the bend.

$$\frac{M}{I} = \frac{\sigma}{y} = \frac{E}{R} \tag{3.2}$$

The second moment of area is defined, where A is the Area, as

$$I = \int y^2 dA \tag{3.3}$$

In order for the beam in question to isolate motion to one plane of bending, the ideal solution would be to use a beam which had a high value of I in one direction but low in the other. From the common beam shapes it can be seen that rectangular section beams fit this description and their second moment of area, about their neutral axis (the centre of the beam), is defined as

$$I_{NA} = \frac{bd^3}{12} \tag{3.4}$$

where the beam has breadth b and depth d. It is clear from this equation that the depth of the beam alters the second moment of area of the beam by a power of 3, whereas the breadth only has a proportional effect. This makes the second moment of area of the beam very dependant on the depth of the beam. If the beam is made with a low depth and relatively high width it will bend easily with a given force.

Take that same force and apply it at 90 degrees to the original force and the beam will be far less susceptible to bending, thus it can be assumed that this will isolate the bending to a single direction, especially as this will also be the primary direction of applied force.

Assuming the beam to be a cantilever with a concentrated load at the tip of its total length l, the bending moment M at any point x along the length of the beam is given by

$$M = -M_A + R_A x \tag{3.5}$$

where R_A is the reactive force at the fixed end of the beam and M_A is the reactive moment. As this is a concentrated load, ignoring self-weight as the beam will be hanging vertically in the test, $M_A = Wl$. The reactive force, therefore, is equal to W, and so

$$M = -Wl + Wx \tag{3.6}$$

The bending moment, elasticity and second moment of area are related by the equation

$$\frac{d^2v}{dx^2} = -\frac{M}{EI} \tag{3.7}$$

where v is the deflection at distance x along the beam from the supported end. Therefore this can be substituted into the previous equation such that

$$EI\frac{d^2v}{dx^2} = -M = Wl - Wx \tag{3.8}$$

and by integration

$$EI\frac{dv}{dx} = Wlx - \frac{Wx^2}{2} + C \tag{3.9}$$

Since, at x=0 $dv/dx = 0$, $C = 0$ and

$$\frac{dv}{dx} = \frac{Wlx}{EI} - \frac{Wx^2}{2EI} \tag{3.10}$$

At $x = l$,

$$\frac{dv}{dx} = \frac{Wl^2}{2EI} \tag{3.11}$$

And if, by further integration of equation 3.10

$$v = \frac{Wlx^2}{2EI} - \frac{Wx^3}{6EI} + D \tag{3.12}$$

and since, at $x = 0$, $v = 0$ by definition, therefore $D = 0$ also and

$$v = \frac{W}{2EI}(lx^2 - \frac{x^3}{3}) \tag{3.13}$$

and ultimately, at the tip when $x = l$

$$v_l = y = \frac{Wl^3}{3EI} \tag{3.14}$$

Now knowing the second moment of area I and the modulus of elasticity of the material E the maximum tip deflection y with load W is known, therefore the elasticity of the beam can be analogised to a spring constant, assuming relatively small displacements and ensuring the beam remains in elastic deformation and does not enter the plastic yield zone where equation 3.14 would not hold true. Using the spring equation, where load W is related to spring constant k and spring deflection Δ

$$W = k\Delta \tag{3.15}$$

equation 3.14 can be re-arranged as

$$W = y(\frac{3EI}{L^3}) \tag{3.16}$$

which corresponds to equation 3.15 to give a spring constant

$$k = \frac{3EI}{L^3} \tag{3.17}$$

Knowing the material to be used, $E = 200 \times 10^9 N/m^2$ and the maximum length, $L = 0.4m$ usable while maintaining good robot arm manipulation (i.e. not working near the maximum extents of joint motion in the desired position for testing), and incorporating equation 3.4

$$k = \frac{3 \times (200 \times 10^9) \times (\frac{bd^3}{12})}{0.064} \tag{3.18}$$

$$k = 781.25 \times 10^9 \times (bd^3) \tag{3.19}$$

Consider also the equation for ω_n of a mass-spring system

$$\omega_n = \sqrt{\frac{k}{m}} \tag{3.20}$$

From this equation and equation 3.19, the approximate effective spring constant, a table of values (Table 3.2) of ω_n was created from the available stock materials and the maximum and minimum masses that would be applied on the manipulator. The minimum value of 0.3kg is derived from the mass of the camera system and the metal plate the camera is bolted to, whereas the maximum weight of 3.5kg is the largest mass that was available for attachment to the bottom of the link.

From the calculations used to create Table 3.2, Table 3.3 was created with additional calculations of k_{norm}, the effective stiffness in the direction normal to the required direction, and from this the stiffness ratio. The stiffness ratio is a measure of the links' susceptibility to vibration in the wrong direction, with higher values representing better vibration isolation. As can be seen from the table, material 2

Table 3.2: Natural frequencies derived from material sizes and masses available.

Material	*Breadth(m)*	*Depth(m)*	$\omega_n\, 0.3kg\, (Hz)$	$\omega_n\, 3.5kg\, (Hz)$
1	0.019	0.005	12.52	3.67
2	0.025	0.005	14.36	4.21
3	0.012	0.012	37.00	10.83
4	0.012	0.006	13.08	3.83

Table 3.3: Effective stiffness of each material.

Material	$k(N/m)$	$k_{norm}(N/m)$	*Stiffness Ratio*
1	1855	26793	14.4
2	2441	61035	25.0
3	16200	16200	1.0
4	2025	8100	4.0

is the most suitable from a stiffness ratio perspective, however with the requirement for a low natural frequency, the chosen material was number 1. This was mild steel bar in a 19x5mm configuration. This was welded to the top and bottom plate of the link as shown by Figure 3.7 earlier. The link was then attached to the robot manipulator, in the orientation shown by Figure 3.8.

The arm was then configured to operate in a translational motion, shown in Figure 3.9, in order to cause inconsistency in the assumed and actual end effector position. Naturally this position error is greater when the optional mass or the manipulator accelerations are increased. From the figure it can be seen that the manipulator will oscillate on step change, or can be made to droop if used in a horizontal position, replicating steady state errors induced by gravity and unknown loads.

In order to ensure the flexible link would not be damaged during step change motions, the maximum deflection to yield was calculated. This assumes that the deflecting end remains stationary while the fixed end is translated. In order to calculate the deflection at yield, equation 3.2 is used with the knowledge of the yield strength of the material $\sigma_y = 350MN/m^2$ and the half-thickness of the beam $y = 2.5mm$. At the point of yield the beam would be under a bending moment of $27.7N/m$, which equates to a load of $69.3N$ at the tip. This can be substituted back into the original tip deflection equation, equation 3.14, to find the maximum tip deflection at yield to be $35.5mm$.

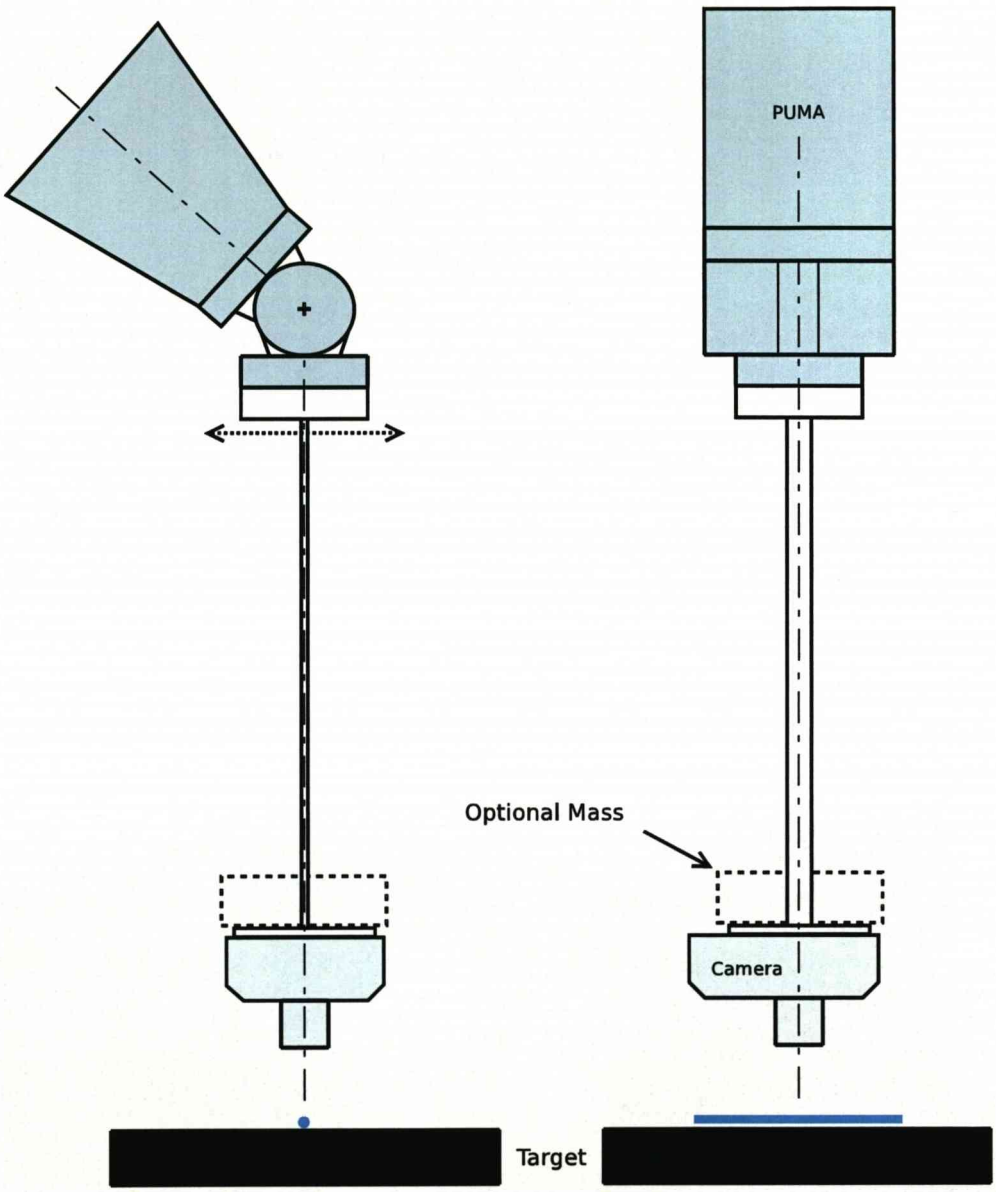

Figure 3.8: The flexible link bolted to the end of the PUMA 560 arm.

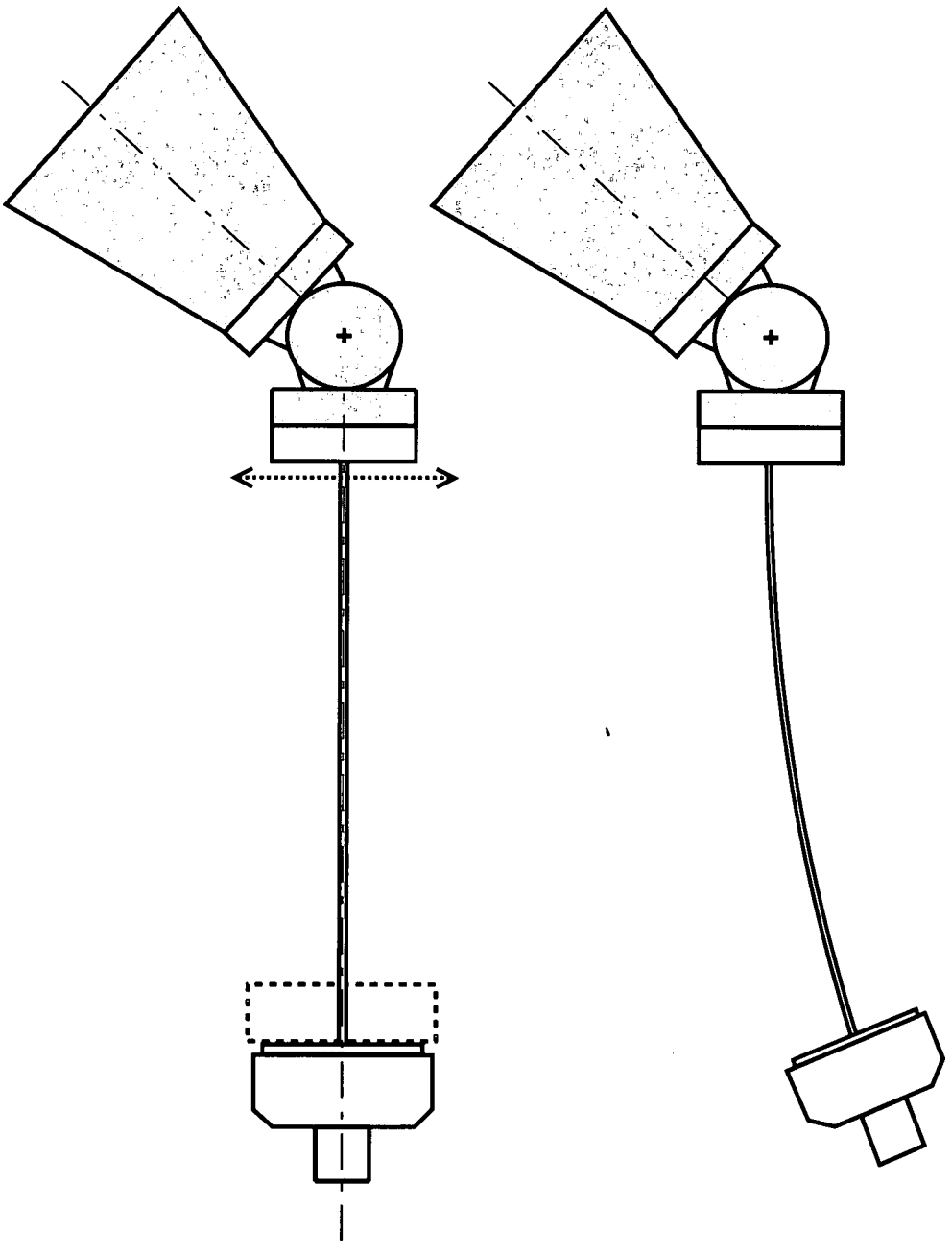

Figure 3.9: Flexible manipulator translational movement.

3.4.1 Verification of ω_n

The mass-spring model used to design the flexible link has some limitations, not least because of the variable centre of mass with the optional weights for the link, but also because it assumed zero damping and ignores compliance in the end plates and joint drive mechanisms. These errors should be small but, for this reason, tests were performed in order to verify the actual oscillation frequency after a positional step change. The results of these tests can be seen in Figure 3.10 and Figure 3.11, from which the frequency of oscillation was calculated. The high value for the 3.5kg mass, Table 3.4, is indicative of the centre of mass being positioned slightly higher than the 0.3kg mass, this effectively shortens the flexible link and increases the natural frequency of the beam to greater than that calculated.

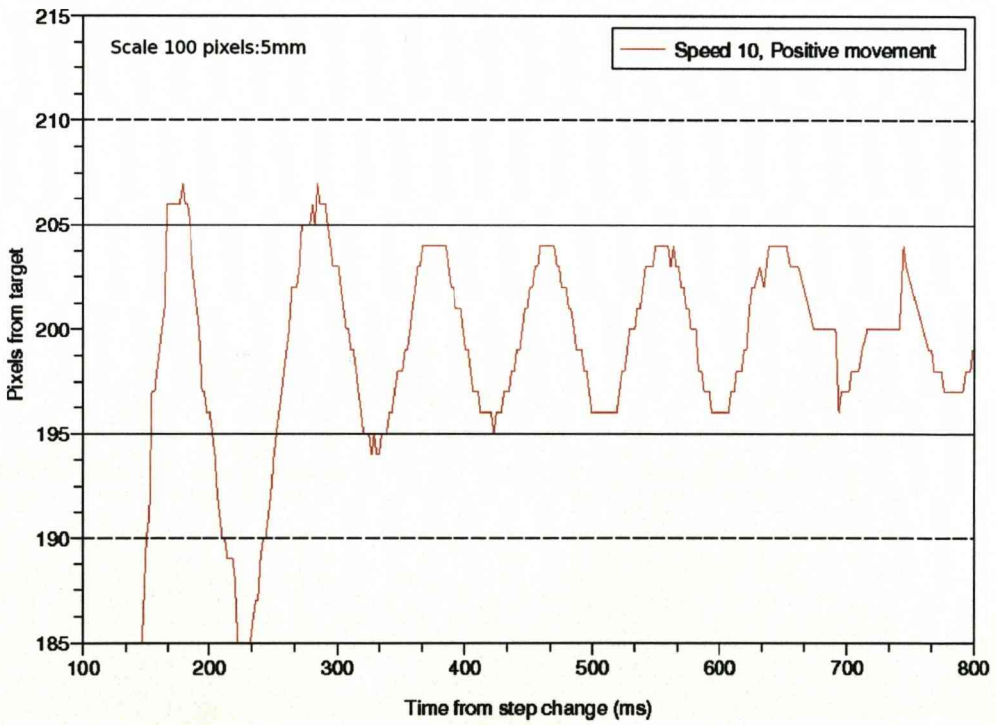

Figure 3.10: Vibrational response of flexible link with 0.3kg mass.

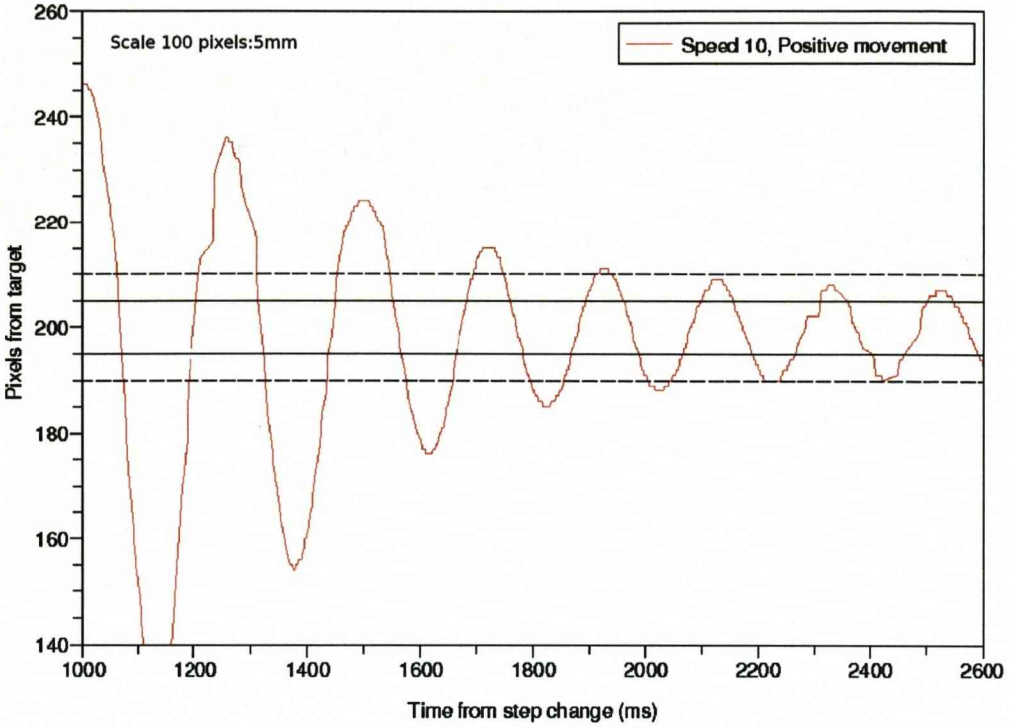

Figure 3.11: Vibrational response of flexible link with 3.5kg mass.

Table 3.4: Oscillation frequencies of flexible link.

$Mass(kg)$	$Frequency(Hz)$
0.3	11.1
3.5	5.0

3.5 Kinematics

This section introduces an area critical to the control of robotic manipulators, kinematics. The kinematics of the robot describe the relationship between joint position and end effector position, with reference to a base coordinate frame. The forward kinematics are used to calculate the end effector position with reference to that frame, in which there will be only one solution. The inverse kinematics are used to find the joint angles required to position the manipulator in a particular pose. Although applicable to robots with n links, the initial diagrams here will show only limited links in order to provide a simplified diagram. There are at least two ways of solving the kinematics of a manipulator, one using trigonometry and one using matrix algebra. Both will be discussed here for completeness. In actual fact, many robots used in industry do not use the kinematic models due to the complexity of their calculation, but rely on simple trigonometric relationships or even simply replay lists of joint positions in order to retrace programmed positions and trajectories. These methods avoid complex modeling and possible errors which could cause damage to the machine or operators, if implemented incorrectly, but cannot account for link compliance.

3.5.1 Denavit and Hartenberg Notation

The matrix algebra solution relies on link parameters as explained by Denavit and Hartenberg [2] as will be described later in the section, however two versions of this parameter assignment method exist - one with the link frame at the distal end of the link (original form) and one with the link frame assigned to the proximal end (modified form). The two are not interchangeable, yet are scattered throughout the literature and both quoted as Denavit and Hartenberg (DH) notation without distinction except in some notable cases, such as [27]. This makes comparison and checking of the kinematic solutions a difficult task, especially as the parameters on

some parallel links, such as links two and three on the PUMA arm, can be combined and placed upon one link only for the purposes of simplification. In the DH notation, links may have prismatic or revolute joint variables, however as this work covers the PUMA only revolute joints are considered variable here. In order to discuss the position of each link, relative to another, a link frame must be assigned to it as well as a description of its parameters.

The DH representation shown in Figure 3.12, courtesy of [51], required definition of four parameters:

1. Link length l_n, the distance between the joint axes along the common normal,
2. Twist angle, α_n, between the joint axes,
3. θ_n, the angle between the links,
4. The displacement along the joint axes, between the links, d_n.

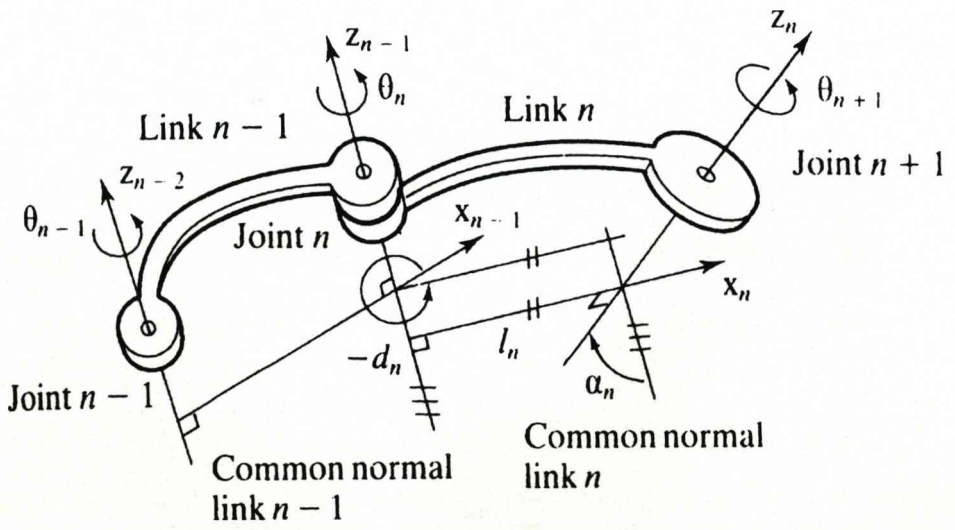

Figure 3.12: Denavit and Hartenberg link parameter descriptions.

These are determined for the robot in question by aligning all joints to zero before assessing the robot and analysing the dimensions given by the manufacturer. Even

on common robots such as the PUMA 560 there is great discrepancy between the values suggested, partially due to differing zero positions and differing approaches to combining parameters for a given application. Certain parameters, such as the outward displacement in Joint 2 followed by a returning inward displacement on Joint 3, are often combined into a single displacement and assigned to one joint. For the case of a static robot this makes little difference to the kinematic solution, but due to compliance in the links and possible problems in object avoidance it cannot always be simplified in this manner.

In order to describe the relationships between the links, a coordinate frame must be assigned to each link, this is where the zero-pose of the robot can affect the signs of some of the DH parameters and result in differing frame assignments between works. The process of assigning links, based on [51], is as follows

1. Number the links from 0 to n, where n is the total number of links and the base link is Link 0.

2. Assign a coordinate system to each link, they are orthogonal and follow the right-hand rule.

3. The base coordinate frame (O) is assigned with axes parallel to the world coordinate frame, and the origin of the base frame is coordinate with the origin of Joint 1, assuming the first joint axis is normal to the world x-y plane.

4. Frames are attached to each link at the joint farthest from the base, therefore Frame 1 is at Joint 2.

5. The origin of the frame is at the intersection of the common normal and the axis of the distal joint. If the axes of the joints are parallel, the position of the frame is chosen to make the distance d_n zero, or a minimum if there is an offset. If the axes intersect the origin is placed there.

6. z axis is coincident with joint axis.

7. The x axis is parallel to the common normal between the joint axes of the link, unless the joint axes are parallel, then the x axis is coincident with the centreline of the link. This is the point where the zero position of the manipulator can affect the link frame system and should be checked to be consistent with the allocation of x axes.

8. The direction of the y axis is defined by the right hand rule.

9. A final frame is attached at the end of the last link at the most distal point, or the point of interest such as between the gripper jaws.

From the general DH notation of the link, a generalised transformation matrix which describes the relationship of one generalised link to another is produced from the product of combined rotations and transformations in the directions described by the parameters. This is usually called an A_i matrix and the application of the specific link parameter information simplifies these to $A_1, A_2 \ldots$ the specific transformation matrix for each link.

The generalised A matrix is

$$\mathbf{A}_i = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & l\cos\theta \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & l\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

Obviously, due to occurrences of 0 and ± 90 degrees in the trigonometric functions, some of the terms equate to zero and simplified matrices are formed as the $\mathbf{T}$ matrices.

$$^0T_i = {}^0T_{i-1}\,{}^{i-1}A_i \tag{3.22}$$

Therefore, in order to determine the end point of the manipulator one must transform all of the coordinate frames from one link to another, the method for doing so

is multiplication of the A matrices to form the manipulator $\mathbf{T}$ matrix, finding this is the forward kinematic solution of the manipulator.

$$^0T_i = A_1 A_2 \dots A_i \tag{3.23}$$

It can be easier to break down the solution into sections, by multiplying out the first three matrices and second three matrices in a six axis robot, then multiplying the two solutions together to get the final solution.

Of equal interest is the inverse kinematic solution, whereby the required joint angles for a particular pose are calculated from the requested position in space. This involves non-trivial mathematics to find the solution. In fact a solution may not exist, or there be many solutions, depending on the robot configuration. The method of solving the inverse kinematics is using the inverse of the transformation matrices. The forward kinematics provide solutions for each cell in the transformation matrix (3.24).

$$^0T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

From these solutions and knowing, for example, from basic matrix algebra that

$$[^0T_2]^{-1}\, ^0T_6 = ^2T_6 \tag{3.25}$$

the inverse of the transform to any frame can be multiplied by the manipulator transformation matrix in order to attempt to single out and solve for the required joint variable. This is not a simple task for a manipulator with a large number of degrees of freedom and often the solutions for a manipulator with $n > 6$ are

indeterminate. In the case of the equations for the PUMA 560, many solutions exist with a positive and negative solution, all of which must be calculated. Often some of the solutions will be rejected due to violations of the maximum or minimum joint limits. The generally accepted method of selecting the correct solution is to select the solution closest to the current position.

The generally accepted zero pose for the PUMA 560 arm is not that of the Liverpool PUMA as modifications have been made for other work with the robot. The actual zero position is shown in Figure 3.13 with a diagram showing the sign convention of each joint. This is clearly not the normal LEFTY or RIGHT zero position that is described in literature. The generally accepted PUMA 560 manipulator arm DH parameters are shown in Table 3.5 and are a concensus found by [27] throughout the literature. They do not agree totally with the dimensions given in Figure 3.14, supplied by Staubli, however various versions of the manipulator diagrams are available online and in manuals, there is surprisingly little agreement between them due to the various versions of the same manipulator; The version is rarely mentioned on the diagrams.

Table 3.5: Accepted PUMA 560 DH link parameters.

Joint	α	a_i	d_i	θ
1	90	0	0	θ_1
2	0	431.8	0	θ_2
3	-90	20.3	125.4	θ_3
4	90	0	431.8	θ_4
5	-90	0	0	θ_5
6	0	0	56.25	θ_6

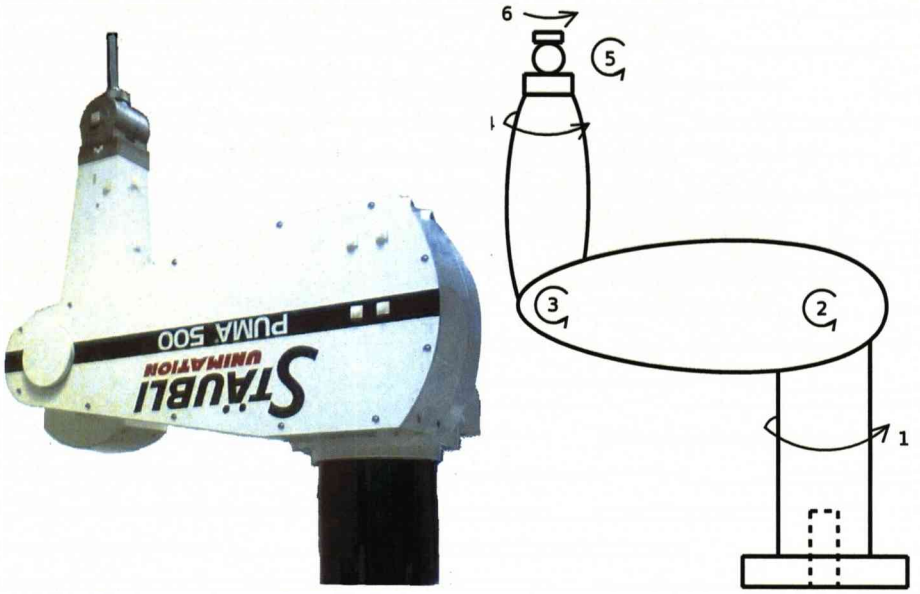

Figure 3.13: PUMA manipulator zero position.

3.5.2 Geometric Solution

The second method of solving manipulator kinematics is using direct trigonometric representation of the manipulator. For manipulators with a large number of degrees of freedom the task of determining the end effector position and pose in space using trigonometry is extremely complex and considered much greater than the algebraic solution. For manipulators with low orders of freedom, this solution can be faster and more efficient to calculate than using symbolic matrix algebra.

The process of determining end effector position on a simplified, especially planar, manipulator with trigonometry is done by assigning a base reference frame and, knowing link lengths, using the angle of each joint to determine the position in space. Figure 3.15 shows a very simple example of this. It is clear to see that when θ is the angle from x_0

$$y_{tip} = L_1 \sin\theta_1 \tag{3.26}$$

$$x_{tip} = L_1 \cos\theta_1 \tag{3.27}$$

Consider Figure 3.16 for a planar three degree of freedom manipulator, position of Joint 3 can be found by

$$x_{joint3} = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \tag{3.28}$$

$$y_{joint3} = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \tag{3.29}$$

and its tip position can then be determined by

$$x_{tip} = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3) \tag{3.30}$$

$$y_{tip} = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3) \tag{3.31}$$

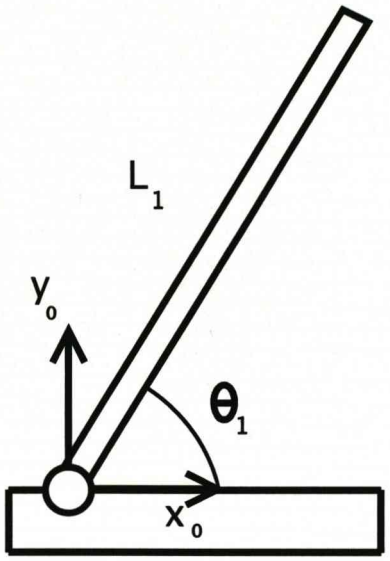

Figure 3.15: Simplified manipulator geometry drawing

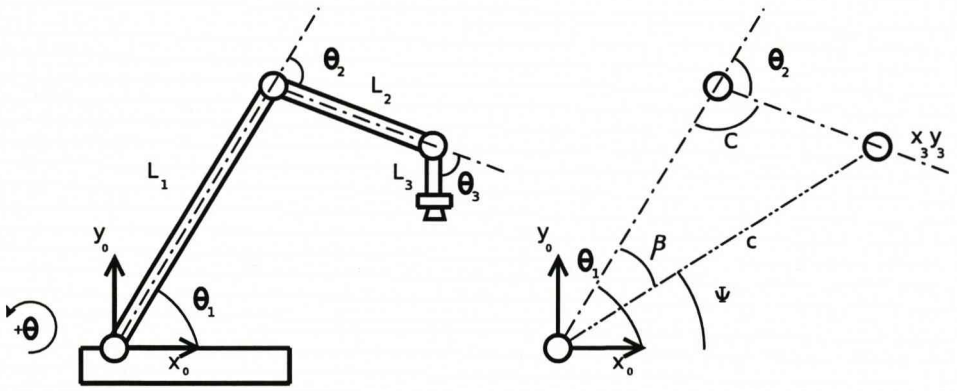

Figure 3.16: 3-Axis planar manipulator geometry

And therefore the pose of the tool is given by

$$\theta_{tip} = \theta_1 + \theta_2 + \theta_3 \tag{3.32}$$

In order to find the inverse kinematics of the same manipulator, the pose required in relation to θ_0, the base frame x axis, is already known and specified. From this and the length of L_3, the position at which Joint 3 has to be located can be determined. Using the Cartesian coordinates of the third joint, x_3 and y_3, against the x_0 y_0 reference frame, the diagonal distance c from the base frame origin can be

calculated from

$$c^2 = x^2 + y^2 \tag{3.33}$$

At this point the computation can determine if the position is within the manipulators range, therefore testing for the presence of solutions, by testing if

$$c < (L_1 + L_2) \tag{3.34}$$

Assuming the test for solutions is positive, using this and the known link lengths of the manipulator, L_1 and L_2, the law of cosines, equation 3.35, can find the angle C between the two links

$$c^2 = a^2 + b^2 - 2ab\cos C \tag{3.35}$$

where

$$c^2 = L_1^2 + L_2^2 - 2L_1L_2\cos C \tag{3.36}$$

therefore

$$C = \cos^{-1}(\frac{L_1^2 + L_2^2 - c^2}{2L_1L_2}) \tag{3.37}$$

From this, the joint angle θ_2 can be found by subtracting C from 180 degrees. In order to find the angle θ_1, use plane trigonometry to determine the angle of side c to the x_0 axis by

$$\Psi = \cos^{-1}(\frac{x_3}{c}) \tag{3.38}$$

and reusing the law of cosines to fine the angle between c and L_1, β, it can be determined that

$$\theta_1 = \Psi \pm \beta \tag{3.39}$$

where the two solutions posed by the plus or minus form the elbow-up and elbow-down options, remembering that θ_2 would need to be re-calculated in order to determine its solution for elbow-down. In any situation the control system would normally select the joint positions nearest to the current positions in order to

minimise the time taken to reach the solution, unless that solution posed a significant limitation on the operating envelope of Joint 3.

In order to test the performance of the flexible manipulator system devised, the PUMA 560 was reduced in degrees of mobility to that of the example manipulator in this section by using the kinematics software to force Joints 1, 4 and 6 to remain at their zero positions at all times. As stated in [52], Joints 4, 5 and 6 are relatively low-torque joints, therefore to allow maximum payload mass options, the use of as few of these joints as possible was considered optimal. With the manipulator in any position with the distal links extended, Joint 1 would have a large inertial load to accelerate. The calculations of joint angles were modified slightly to take into account the zero angles of each joint not being as described in the previous section, namely Joint 3 being 90 degrees displaced from the idealised model presented above. Using inverse kinematics, the manipulator can be positioned anywhere in the world $x - z$ plane.

A fixed z position was chosen relative to the platform that the manipulator was located on, keeping the camera lens within an optimum position to maintain good focus and sufficient illumination of the scene to generate a stable image. The control loop, detailed in Chapter 5, repeatedly sampled the joint encoder data, to determine the position of the robot using the forward kinematics. The camera data was then used to determine the error from the target and was combined with the current rigid-body assumption of position in order to adjust the *demanded* position of the manipulator to the correct position, irrespective of where the rigid body kinematics suggest the tool is. Within the magnitude of flexibility allowed in the test system, as explained in Section 3.4, the flexibility can be considered to be linearly related to the effective spring constant of the link and force created by the acceleration of the payload.

3.6 Summary

In this chapter an introduction to robot manipulators was given and a review of manipulator types was presented. Discussion of the advantages and disadvantages of each type explained the usual applications of each and introduces consideration of the effect of link compliance in the accuracy of the robot. The requirements of a potential test-bed robot for this research was given, including specification on backlash and rigidity to ensure the compliance being examined was that intended and not some unknown value.

The selected testbed was presented and details of its parameters and capabilities were given including joint encoder accuracy. The hardware controller design was explained and the interface method used to control the hardware was summarised, but is explained further in 5. The manipulator required performance degradation in the form of a compliant link, in order to test the visual control system. This was selected as a compliant link fixed to the tip of the manipulator and the details of its design and performance testing were given in subsequent sections.

Finally the methods of solving robot kinematics and inverse kinematics were presented, both in matrix algebra and in geometric solutions, as applied to the robot in this research. Specific details of and constrains on the application of these solutions to the PUMA used in this research were discussed at the end of the chapter.

Chapter 4

Vision Systems

Visual input for computer systems began in the late 1960s and early 1970s, both Complementary Metal Oxide Semiconductor (CMOS)-based and Charge-Coupled Device (CCD)-based technology being developed simultaneously. The CCD was invented in 1969 by Willard Boyle and George E. Smith at AT&T Bell Labs and by 1970 they were able to extract linear image data.

During the same period Noble [53] and Chamberlain [54] were developing CMOS-based imaging. Originally the CMOS systems were of lower quality, suffering high noise, low speed and poor scalability - original test systems were around 10 x 10 pixel arrays. In more recent years the gap in quality and price between the sensors has reduced, as predicted in work carried out by Carlson [55]. The application described in this thesis leads to a choice based primarily on speed and adjustability, not quality of image, although that quality is important as will be explained later. Megapixel sensors have been available since since the late 1980's, however only in the last 10 years have CMOS and CCD cameras reached speeds capable of high speed real-time machine vision with higher resolutions [56]. Very high frame rate sensors are currently in production; *Vision Research* [57] produce a CMOS-based camera capable of 190,476 frames per second in reduced resolution mode, and over 6,688

frames per second in its maximum resolution of 800x600 pixels. These are still very expensive, but this only shows how rapidly the technologies are progressing when compared to those published only eight years ago. These were either low speed and high resolution, or high speed and low resolution [56, 58]. High speed machine vision is now used in a variety of automated processes, from high speed pick and place operations, quality inspection on production lines and vision controlled cutting processes [59, 60, 61, 62, 63]. More specific to this work, Smith [64] uses visual input to control weld-pool size and weld penatration in real-time, during the TIG welding process. This work demonstrates the ability to measure the geometric properties of the weld during the molten phase and subsequently control the processes processes by visual input in order to deliver the required weld bead size. The work also highlights the need for good contrast between the molten weld pool and the workpiece being welded. This work also produces a novel edge feature correlation algorithm for real-time vision based molten weld-pool measurements, as published in the highly regarded Welding Journal [65]. Robotic welding largely employs vision systems [66] and robotic welding [67] for automated assembly and inspection, its is regarded as proven technology in this field with applications in ship building [68], underwater engineering [69] and seam tracking [70, 71]. Some systems derive their measurements from the image alone, but the more common method uses a laser line projected onto the surfaces being welded to profile the materials being welded, whereas some utilise simple process lighting or the light produced by the welding process itself [70]. Li and Zang [72] even measure and control arc lenth and energy by the wavelenth of the radiation produced during the weld process.

CCD Based Sensors

A CCD is essentially an analog shift register, enabling analog signals to be transported through successive stages, regulated by a clock signal. Charge coupled devices can be used as memory or for delaying analog sampled signals. They are most

suitable for serialising parallel analog signals, such as arrays of photoelectric light sensors, as seen in Figure 4.1.

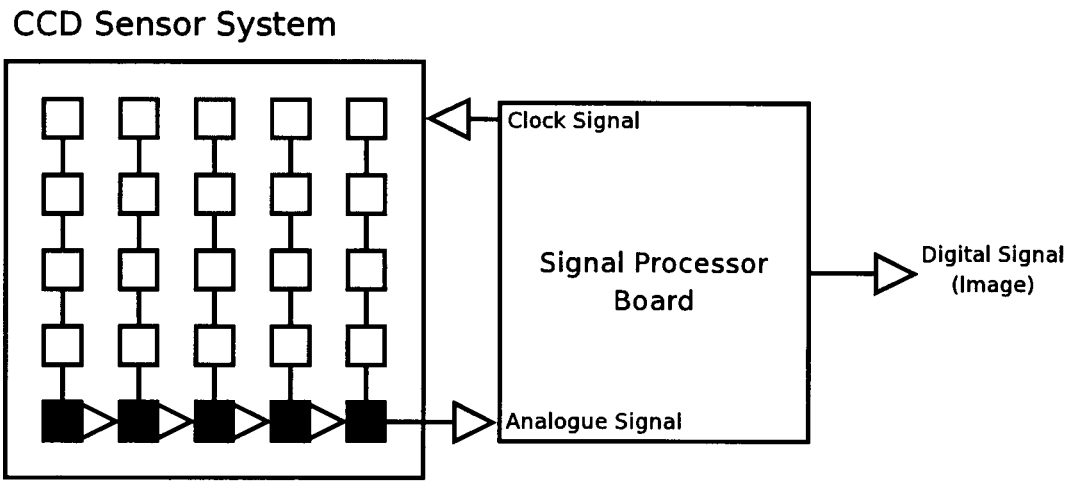

Figure 4.1: A diagrammatic view of CCD image extraction.

An image is projected, through a lens, onto an array of capacitors. The photons charge the capacitors in a manner that is proportional to the light intensity at that site. Once the image has been captured, each capacitor passes its charge onto its neighbour except for the end capacitor which passes its charge on to an amplifier system and in turn onto the sampling and digitising processor, then on to memory. When this process has been repeated for however many capacitors are present in each column, the whole image has been processed and the CCD is ready for exposing again.

Clearly, to see any small section of the image the system would have to process most, if not all, of the array in order to gain access to the required section. This means that even if the requested area is only 100x100 pixels, the whole megapixel sensor must be scanned, digitised and transferred to memory before it can be used. Some modern CCD cameras process segments of the CCD in parallel in order to allow windowing (see Section 4.2.1) and faster image acquisition.

CMOS Based Sensors

As shown in Figure 4.2 the pixels are addressed by both a row and column access system, with each pixel being constructed with three transistors and a photodiode, Figure 4.3. Incident light causes an accumulation of charge on the photodiode, in turn creating a voltage change that is related to light intensity on that pixel.

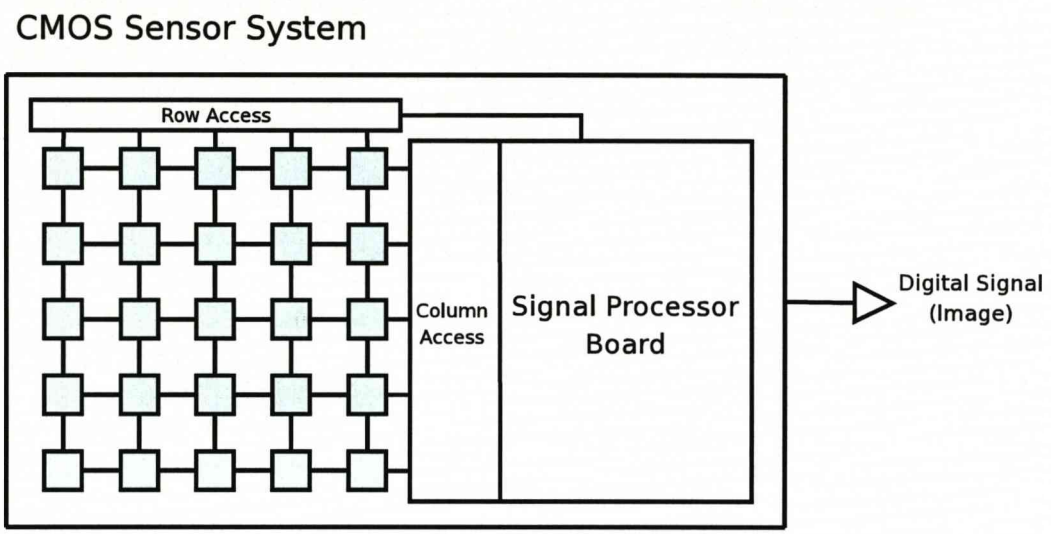

Figure 4.2: A diagrammatic view of CMOS image extraction.

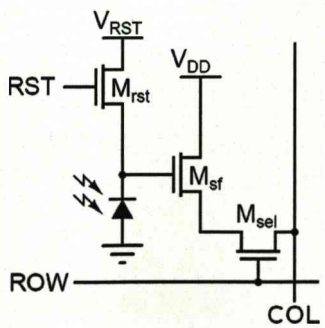

Figure 4.3: The three-transistor CMOS pixel implementation.

One transistor, M_{rst}, acts as a switch to reset the device and when this transistor is activated, the photodiode is connected to the power supply, V_{RST}, thereby clearing all charge. The second transistor, M_{sf}, acts as a buffer, an amplifier which allows the pixel voltage to be observed without removing the accumulated charge. Its drain, V_{DD}, connected to the power supply, is typically coupled to the drain of the reset transistor. The third transistor is the row-select transistor, M_{sel}. This switch allows a single row of the pixel array to be read by the digital processing electronics.

4.1 Image Formation

As with all cameras, computer vision cameras require a lens in order to focus the light before it reaches the CCD or CMOS sensor plane. A simplified diagram of the key features of a lens system are shown in Figure 4.4.

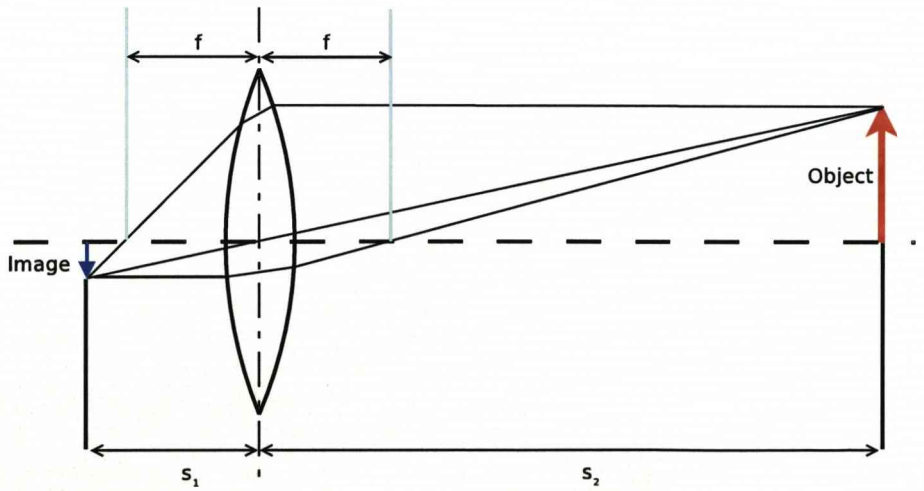

Figure 4.4: Thin lens optics, simplified diagram.

Lens variables are related by the thin lens law:

$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f} \tag{4.1}$$

Where f is the focal length of the lens and S_1 is the distance from the lens to the image plane, and S_2 is the distance to the object. For objects at a distance further

than the focal length of the lens a real inverted image is formed at a distance behind the focal plane of the lens. When an object being imaged is at infinity the inverted image is produced at the focal length of the lens - this is where the imaging sensor would be placed if a distant object were being observed. For objects closer than infinity but more distant than the focal length of the lens, the lens must be moved forwards away from the imaging sensor in order to bring the focus point back to the plane of the imaging sensor.

The image size on the sensor, is related to the object size by the magnification, which is derived by:

$$M = \frac{S_1}{S_2} \tag{4.2}$$

Therefore, for objects much more distant than the focal length of the lens as is the case in most camera situations,

$$M = \frac{f}{f - S_2} \approx \frac{f}{S_2} \tag{4.3}$$

This means that, for reasonably distant targets, image size can directly represent the size of the object via the magnification equation. In turn this means that at twice the distance from the camera, the same object will appear half the size. This can be used, knowing the lens specification, to determine the distance of a known-size object in relation to the camera - this is useful in visual servoing as it allows the robot control system to identify its position in world coordinate space, based purely on a calibration object or mark.

4.1.1 Digitisation

Common to CMOS and CCD technology, as well as all digitised images by definition, is the principle of a pixel. A pixel is representative of the light intensity at a particular location in an image. Take, for example, the continuous real image $I(x, y)$. When this is digitised by an imaging system such as a CCD or CMOS camera, the

image falls upon the sensors photo-sensitive sites and is converted into a discrete array $I'(j,k)$ of pixels, where j and k are the coordinates of the photo-sensitive site responsible for the each pixel. Figure 4.5 explains this visually. Figure 4.5 is

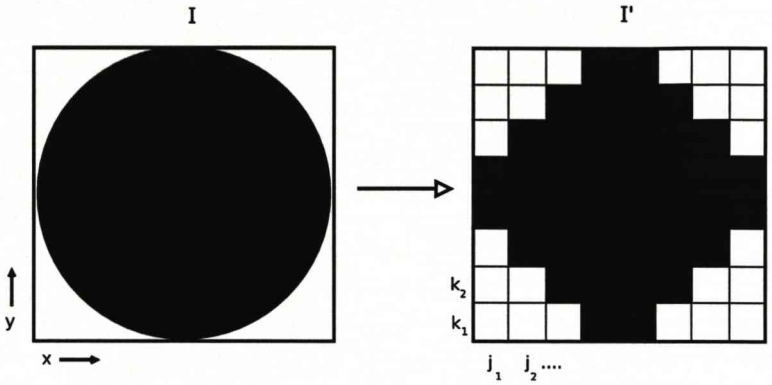

Figure 4.5: Digitisation of a real image.

representative of a binary simplification, modern machine vision cameras are either colour or greyscale. In the case of a greyscale camera, each pixel records the intensity of light present at each photo-sensitive site and converts it into a digital value on a scale of 0 to 255. This is known as the pixel depth - most greyscale cameras work at eight bits per pixel allowing up to 256 different shades including black and white. This allows the image to show shading instead of a simple binary representation, allowing texture detail or objects to be identified in the image by areas of similar values.

Likewise, in a colour camera system, the array of photo-sensitive sites is organised into a pattern where each photo detector has its own colour filter fixed above it, allowing only red, green or blue light to pass. These pixels are arranged in a regular pattern and this allows the captured image to be manipulated by a "demosaicing" algorithm, reproducing the other colour intensities at each pixel. There are many filter arrangements, the most common are Bayer and CYGM, Figure 4.6. A high quality sensor provides a pixel depth of twenty four bits per pixel, providing capabilities of up to 16.77 million colours for each pixel. With increasing colour capability, the required storage size for each image increases -

doubling the pixel depth doubles the image storage or transportation size, assuming no compression is used.

4.1.2 Pixel Density

The size and relative proximity of each pixel on the sensor surface defines the pixel density of the sensor. Assuming all other factors remain constant, higher pixel densities, provide higher image resolution after digitisation. Figure 4.7 shows the same process as Figure 4.5 but with a sensor with twice the number of pixel sites in each axis of the image. Doubling the number of pixels in each axis provides 2^2 times the total number of pixels, multiplying storage and transport overheads by 4. This also increases image resolution and allows the original object to be digitised in a way that is more representative of the original than the lower density sensor allows. This allows the computer vision system to extract more information from the image, including more accurate edge detection and so create better representations of their target. When considering edge detection, or analysis of thin lines in a digitised image, the system designer must be aware of the effect of aliasing. The fine detail capability of the sensor system is limited to the size of a single pixel in the image. Should a detail be less than the size of a pixel it may not be measured at all. Conversely, one that is just larger than a single pixel width could also "grow" depending on its light intensity and appear to be nearly twice as large as it actually is. This can manifest as pixel jitter, with the output oscillating back and forth

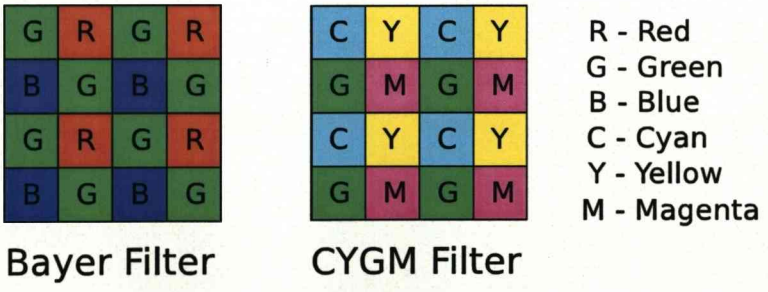

Figure 4.6: Two common CCD sensor colour filter arrangements.

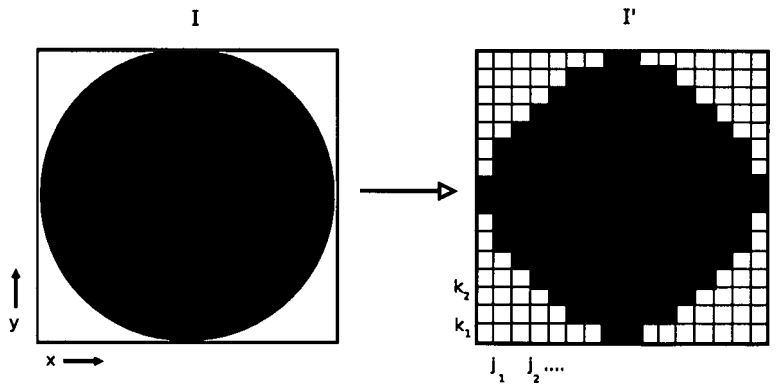

Figure 4.7: The effect of increased pixel density.

between two adjacent pixels despite no oscillation actually occurring. Any control loop should be robust enough not to be affected adversely by such pixel jitter.

4.1.3 Exposure Time

It would seem natural to assume that, between frames of video, the sensor is exposed to light for the whole duration of the frame. Although physically there is no shutter in digital machine vision/motion cameras, there is effectively still an exposure control. The length of time that the photo-site is allowed to accumulate charge from the photons is varied, longer collection time allows the accumulation of more charge and therefore produces a brighter image. There are two areas where the exposure control can be limited - very high contrast images and very high frame rates. With a very high contrast image, such as a general scene with a very dark area situated somewhere in view, or a largely dark scene with a small, lighter area. One exposure setting will not suit all of the image; in the first case, the dark area will not produce sufficient phonic activity to accumulate enough charge to register on the sensor, leaving an area of lowest intensity which is devoid of detail. In the second case the light area may be over-exposed, saturating the photo-sites with light, leaving nothing but a large patch of image with the highest intensity value. Both cases lose detail and information from the image.

The second problem arises when frame rates increase significantly. The length of time required to accumulate charge in darker situations may be relatively long, when added to the digitisation and transport time for each frame, attempting to view a darkened area with a high frame rate the exposure time required may be longer than the framerate requested. As a consequence the framerate of the camera will reduce in darker situations, unless a larger lens aperture or a higher sensitivity sensor is used. If a larger lens aperture is not available the system may use process lighting, whereby the area of interest is purposefully lit in a way that provides the optimum lighting and detail recognition. Another option is to create an illuminated target, whereby the target is brighter than the surroundings and the surrounding detail is irrelevant.

4.2 PixeLINK A641

Having reviewed the literature on the cameras available for this experiment, the camera chosen and presented in this thesis is the PixeLINK A641, Figure 4.8, CMOS based, machine vision camera. This is a monochrome camera connected to the host computer using a Firewire (IEEE 1394) cable; a high speed serial link. The camera is capable of 1.2 megapixel resolution (1208x1024) and delivers this in an uncompressed format at 14 fps, losing no data upon transmission although the controlling machine must be capable of processing at that framerate. In order to obtain best results, the camera is capable of auto-adjusting its exposure time and altering its signal gain in order to provide proper imaging in low-light conditions.

The A641 is supplied with with a software developers kit (SDK) including full API documentation and drivers for the Windows^{TM} operating systems, unlike other cameras available at the time. The primary reason, however, for this choice is the ability of the A641 to "Window" to increase framerate.

Figure 4.8: The PixeLINK A641 monochrome machine vision camera.

4.2.1 Windowing

The most interesting feature of the A641 is its "windowing" ability. Unlike some digital video imaging systems such as most CCDs, see Section 4, the A641 is capable of selecting only a certain part of the CMOS sensor to use. This can vastly reduce the amount of pixels scanned, and therefore reduce the quantity of data produced. The overall resolution remains the same in that area but the frame size is reduced. This means an area of interest can be singled out, or rows at least - depending on the implementation of the digital processing hardware and memory. The consequence of the reduced pixel scan quantity is the ability to significantly increase the scan rate and so the frame rate of the camera.

Windowing in on a very small section of the image, comprising only 16 rows and 24 columns, the A641 can obtain framerates of nearly 900 fps. The framerate is decreased roughly proportional to the increase in area scanned, with some special exceptions; assessing the framerate of the camera with a fixed number of rows and varying column dimensions it becomes clear that the image can be broken down further than just rows - if the window is less than half the width of the total sensor width the speed is increased substantially again. It seems likely that this is due to the

A641 processing rows in two halves, possibly to enable these high speed windowing features, but equally likely is that this allows full-image processing to be done in multiple parallel processes.

4.3 Windows^{TM} Host Computer

PixeLINK do not produce a Linux driver for the A641, unlike their newer models. The A641, therefore, would have to be controlled by a computer with the Windows^{TM} operating system. The A641 uses proprietary control commands and data structures; the manufacturer was unwilling to release this information, leaving no option to create a Linux driver. For these reasons alone, the camera was controlled by a standalone Windows machine, with the data transferred to the robot-controlling machine via a dedicated 100 Base-T ethernet link.

If the network link were working at its theoretical maximum it should be capable of transferring 100 million bits per second. Working at full resolution the camera would need at least 146.8 million bits per second of bandwidth:

$$bps = I_r \times D_p \times Framerate \tag{4.4}$$

$$= (1280 \times 1024) \times 8 \times 14 \tag{4.5}$$

$$= 146.8[Mbps] \tag{4.6}$$

However the overheads added by the encoding of the frames so that they can be decoded would increase this further. This is a far higher transfer rate than is possible with 100 Base-T transmission. This, therefore, requires the image processing to be carried out on the host machine and only the required information extracted from the image.

4.4 Image Processing

Image processing is the creation of a modified image from an original, that is somehow more useful for the task than when it was originally captured. Many examples of image processing can be seen on commercial photograph editing software, where the user can adjust colours, brightness and contrast to create a more ideal image, or to remove inaccuracies in the image colour and match it more closely to reality. Some of the image processing tools in these packages are capable of visually complex operations including warping, medium changes (to make an image look like a painting) and embossing. These are rarely useful for anything other than visual enhancement for a human viewer - machine vision usually requires simplification of an image. Representation of an image in a simpler, clearer form allows more rapid assessment of the contents of that image, which is often vital for the limited processing power available in a computer system.

4.4.1 Thresholding

The image viewed by the A641 used in the experimentation provides a large amount of information, the vast majority of which is not required in this research. In order for the camera subsystem to determine where the end effector of the manipulator lies, it requires a clear and simplified view of the scene. The slight variations of the background area greyscale values and the blurring at the edge of the target are inconsequential to the operation of the system. However, they make the process of assessing the image more complicated and tend to slow the processing on a low power computer. Thresholding is the simplest form of image segmentation, in order to simplify the image and to prepare the image for pattern recognition. This is the process by which a value of pixel brightness is selected, either arbitrarily or by some process such as histogram analysis of the scene. All pixels at this intensity, or brighter than this threshold are changed to full brightness (white in the case of the

A641 monochrome image), all pixels below to full darkness (black). If the threshold value is selected properly the object will be highlighted as a block representation in this new binary image and the background detail completely removed, see Figure 4.9. Threshold value selection has been the subject of several studies, which have used differing methods to select an appropriate value for a specific application. A thorough overview of these studies, [73], details the various techniques according to the information being exploited. These can be narrowed into six main areas:

1. Histogram shape-based methods, the physical shape of the image histogram is considered, a threshold value is chosen using some combination of rules applied to the histogram.
2. Clustering methods, grey level samples are clustered in two parts.
3. Entropy-based methods, using algorithms that calculate the entropy of sections of the image in order to determine which should be foreground and background.
4. Object Attribute based methods, where a determination of similarity between grey scale and binary images is used to select the threshold.
5. Spatial methods, using high order statistical and probability distribution.
6. Local methods, these use an assessment of the area surrounding each pixel in order to determine whether it should be high or low valued.

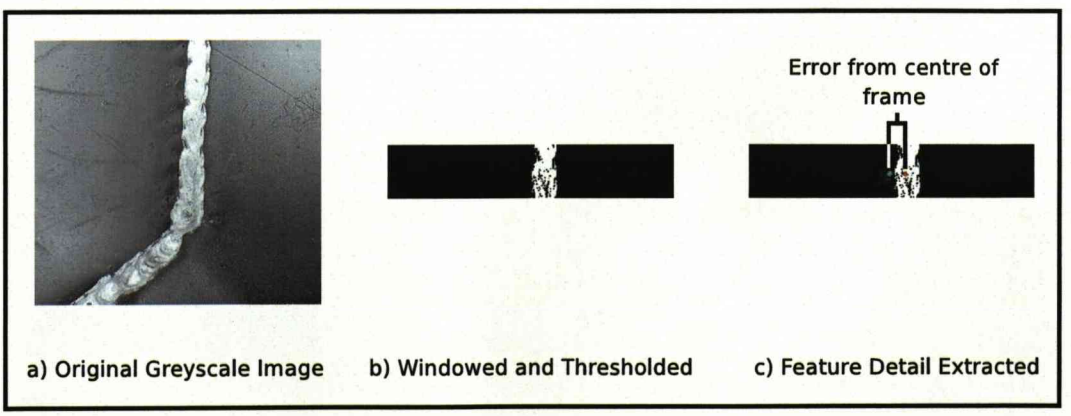

Figure 4.9: An example of thresholding applied to a welding process image frame.

As the target in the experimental setup could be artificially chosen as a high-contrast target, complex thresholding techniques were not required and an arbitrary value was chosen, as explained below, after the A641 gain and shutter length was selected automatically by the camera. The target in the experimentation was a pair of cold-cathode lighting tubes on a matte-black background. These were sufficiently high contrast to mean that a threshold value around 128 (of 0-255) was perfectly acceptable for target segmentation. Although the target tubes used in the tests were of slightly different brightness, this was made irrelevant by the pattern recognition system used - so long as the target had some continuous part above the threshold brightness level its central point would be identified. The thresholding was applied on a pixel by pixel scan of the frame array on its arrival at the host computer. This kept the pre-processing of the image down to a minimum and allowed the host computer to maintain a high frame rate while extracting all the information needed.

For use in a final production version of this experiment, a more complex thresholding process could be required, however it would be operating on a more state of the art computer system, or could be operating from an FPGA module designed solely for the process of conditioning the image before its introduction into the control system. White and Rohrer [74] used locally adaptive thresholding system based on local contrast which would be suitable and would be a starting point of futher experimentation. White discusses the use of a structuring element whereby an eight by eight pixel cross formation is used to determine the surrounding pixel greyscale distribution and apply a threshold to the central pixel based upon those local pixels. This requires significantly more processing power than a simple fixed level, image-wide threshold but is not as intensive as the statistical analysis methods which are better suited to single image detailed processing instead of real-time control.

4.4.2 Pattern Recognition

Pattern recognition is the process by which parametric information about the objects portrayed in a frame, or stream of frames, is identified, extracted and acted upon. The area has been subject to detailed research and falls across the boundaries of machine vision, machine learning and intelligence, areas of increasing interest with the ongoing research into autonomous vehicles. Object and therefore pattern recognition were among the initial reasons for the development of machine vision and the techniques used have progressed significantly.

As with image processing, pattern recognition used within the scope of this thesis was not required to be detailed, in fact the primary concern was to get fast, yet accurate, results from the image stream presented to the host computer. This was done by taking the thresholded image and marking the centre of the windowed section, then scanning the centre line of the image until a gradient was detected. This point was recorded and the scanning continued until a second change in gradient was observed and recorded. The width and centre point of this section was then recorded and the difference from the centre of frame determined.

This gives a single value representation of the error from centre of frame which can be transmitted at very high speed across a standard network connection to the controlling computer system. Although this does not take lens effects or camera models into account, this was not the purpose of the camera subsystem. Since extra calculations could be carried out on the error data at a remote computer, the requirement for maximum possible framerate was prioritised and the task of interpreting the error data was handed to the faster, real-time Linux machine.

4.5 Target

The target used is of significant importance in visual servoing, it is capable of providing information such as pose, range and overall position to the control system. In order to deliver this information accurately the target may be used as a one-off calibration before starting a process, or the target may be part of the process, such as line following in the welding process. With known variables such as line thickness and with the possible addition of "helper" target items the camera can calculate its range from the target and a good estimation of pose is possible by investigating the relative distance and size of the additional targets.

Consider the example of repeated circular dots along either side of a line that is being followed by some flexible and uncalibrated manipulator. From the distance between the dots and determining whether the dots are the same size across the image, the camera can identify the range from the target as well as the angle of approach, while following the central line as a position reference.

Due to the possible problems caused by exposure control, section 4.1.3, the target used in this research was a high contrast target - see Figure 4.10. The diagram shows the PixeLINK camera (A), the blue cold cathode tube viewed from its end (B) the matt black base (C) used to prevent reflection. The main parameter of interest in the line target being followed by the pattern recognition system was the overall central point, although more information *could* have been extracted for future tests. The brightness of the tube emissions vary as the tube warms, the target tube was allowed to warm for five minutes prior to testing, allowing the emissions to stabilise. A method of switching between two tubes was created, however a software solution to this problem was later used as a more efficient solution.

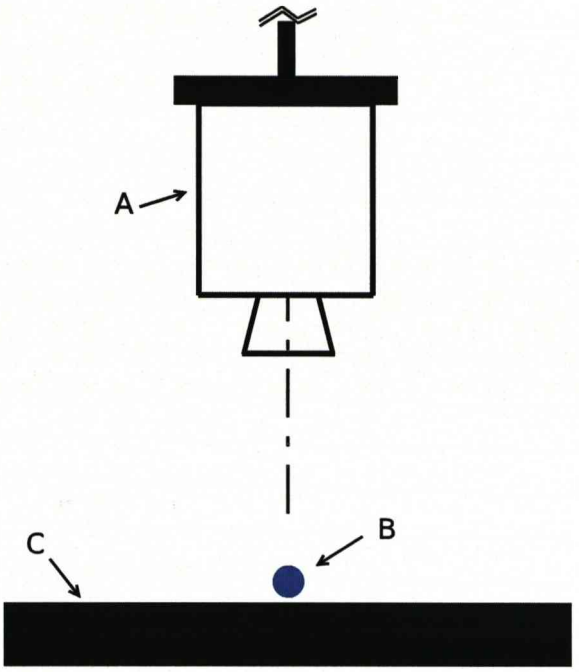

Figure 4.10: Cold cathode target arrangement.

4.6 Summary

In this chapter an investigation into the machine vision hardware available was completed, concluding that the PixeLINK A641 CMOS camera system was most appropriate for use in this research. The A641 was shown to have features that allow very high frame rates, including the ability to "window in" on a specific section of the sensor area. This camera also has a clear programmer's interface documentation and developers kit in order to allow rapid development of the software required to control the camera. The software to monitor the camera output and extract the relevant data was investigated and thesholding and high speed edge detection algorithms employed to calculate the positional error from target at a framerate of at least 350Hz when connected to the control computer.

The target used in the investigation was detailed, using a cold-cathode lamp, providing a high contrast line for the software to follow, although in practice the software was sufficiently robust as to follow a simple line on paper in ambient

lighting. The cold-cathode target was employed to ensure the target did not change characteristics during the experimentation.

Chapter 5

System Implementation

This chapter describes the implementation of the complete system developed in the previous chapters. It discusses the details of interfacing the separate sections and the effect the implementation has on the overall system effectiveness, including the control strategies employed. Very little information is available in literature about specific system implementation with most existing work being explained from a theoretical perspective only; generally developed on existing platforms requiring little modification or presented as simulated results.

5.1 Interfaces

As described in Chapter 1, visual servoing of a manipulator involves several key areas including vision hardware, manipulator hardware and a processing system, the controller - all of which are areas of research in their own right, as explained in earlier chapters. This section explains how the various parts of the system interface with each other in order to combine to form an overall system. Each subsection is a description of the actual interface method and will later be explained in the wider context.

5.1.1 Slave Interface

In order to allow complete control over the manipulator the little-used SLAVE interface was chosen. This is a joint angle control interface at a much lower level than the Alter command. When attempting to control the joint angles, accelerations and forces directly, without relying on the original VAL controller, direct access to the joint encoder information is needed. This is provided over a serial RS-422 connection.

RS-422 is a standard, industrial, serial protocol - not normally used on normal desktop computer systems, therefore it was necessary to use an RS-232 to RS-422 converter - this is a simple piece of hardware that adapts the signal voltages and is transparent to the devices at either end.

Slave-to-PC Communications

The data provided by the controller includes all joint angles, the number of joints, error status and robot gripper state (open or closed). This information is encoded into a packet of data in the form shown in Figure 5.1. This packet has a fixed structure but variable length due to byte stuffing. When the packet is created it is made with a header and a footer section, with the data embedded between them. The header and footer always contain the same characters, except the check-byte in the footer, in order to allow the communications software to determine where the packet begins and ends. This is one of many methods of denoting the start and end of a packet, another includes a header that contains data telling the software how many bytes are in each packet. The problem with this method is that the

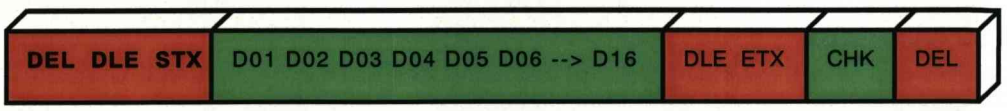

Figure 5.1: A Basic SLAVE packet.

data field could, in theory, contain the footer as pure coincidence. To avoid this the system employed by the SLAVE control software is to "byte-stuff". Any coincidental occurrence of the first footer byte within the datafield should *always* be followed by a duplicate byte, this allows the software to determine between the footer and a random occurrence of the same bytes in order. This second byte is not included in any checksums and is discarded immediately.

Within the packet of data there exists two subsections, the first four bytes contain information regarding the state of the robot gripper, the error status of the controller and the number of joints available. The remaining twelve bytes, assuming no byte-stuffing occurs, are pairs of bytes that represent the joint angles starting at the waist, Joint 1. Each joint angle is provided in two-byte words, low-order byte first and represents a signed sixteen bit integer when re-ordered and concatenated. As shown in Figure 5.2, the bytes are bitshifted and merged to create the sixteen bit word, 0 to 65535 for each joints' 360 degree theoretical range. Using the normal binary sign convention the most significant bit represents the sign of the word, allowing each joint to use 32,768 points to represent each 180 degree sweep away from 0, central. This is true of the first five joints, although they cannot necessarily reach the full ± 180 degree range due to mechanical constraints. Joint 6 is different due to being able to rotate more than 360 degrees in total, in this case the word represents 720 degrees rotation. Clearly this determines that the step resolution on this joint is half of that of its counterparts, however the data can be treated in the same way, but with a differing scale factor in calculations.

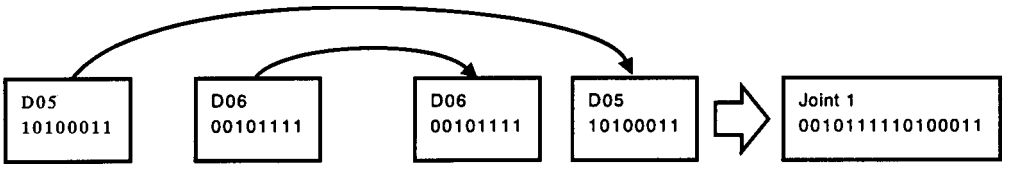

Figure 5.2: Generation of joint angles from packet bytes.

The main process of communicating with the SLAVE interface is carried out by a hard real-time interrupt service routine (ISR), owned by the RTAI hardware abstraction layer (HAL), and is initiated by inserting the main module into the linux kernel. This sets up the initial realtime environment; Marking the main trajectory control process ready to execute, setting up the inter-process communication (IPC) mailboxes (MBX) and shared memory areas (SHM), as well as their resource protection semaphores. The ISR itself is executed at every single byte in order to decode the packet byte by byte and allow immediate response after the first byte of the second packet is received. The first packet is not replied to and is used to get the start-up position of the manipulator.

The next packet is replied to with a copy of the first received packet, in order to initialise and stabilise communications without robot motion. From this point on the ISR monitors two mailboxes, the control threads mailbox and the commandline mailbox, see Section 5.1.4. If new data is recorded in either of these mailboxes, it is converted to a new target and the joint controllers attempt to move the joints to their new target position. If these mailboxes do not contain new target data, the ISR simply uses the last known position in order to maintain the joint locations and not break the communications timing requirements.

5.1.2 Camera Link

The camera used to execute the EIH visual feedback was the PixeLINK A641, as described earlier. This is a machine vision camera which works in 8 or 10 bit greyscale with pixel resolutions up to 1280x1024. It transmits the images over a Firewire cable using proprietary drivers and its own API. Unlike most normal cameras, the A641 is capable of "windowing" - a process by which only a selected area of its imaging surface CMOS is read. This allows a much smaller amount of data to be processed on the camera, which in turn allows much faster frame rates to be achieved. Theoretical frame rates of up to 900 frames per second (fps) are possible with a very

small window, however the smaller the window, the lower the amount of useful information that can be retrieved from the image. The Firewire communications link is capable of a theoretical 400Mbps and is physically very similar to a USB link. The maximum length of the firewire cable is limited, unless very high quality cable is used the standard length of cable is 2m, up to 5m are available. The equipment used in this research utilised a 2m cable, with careful positioning of the camera host computer this is all that is required for the robot to work anywhere within its operating envelope. Figure 5.3 shows the PixeLINK camera mounted to the end of the flexible link directly above the target, during testing.

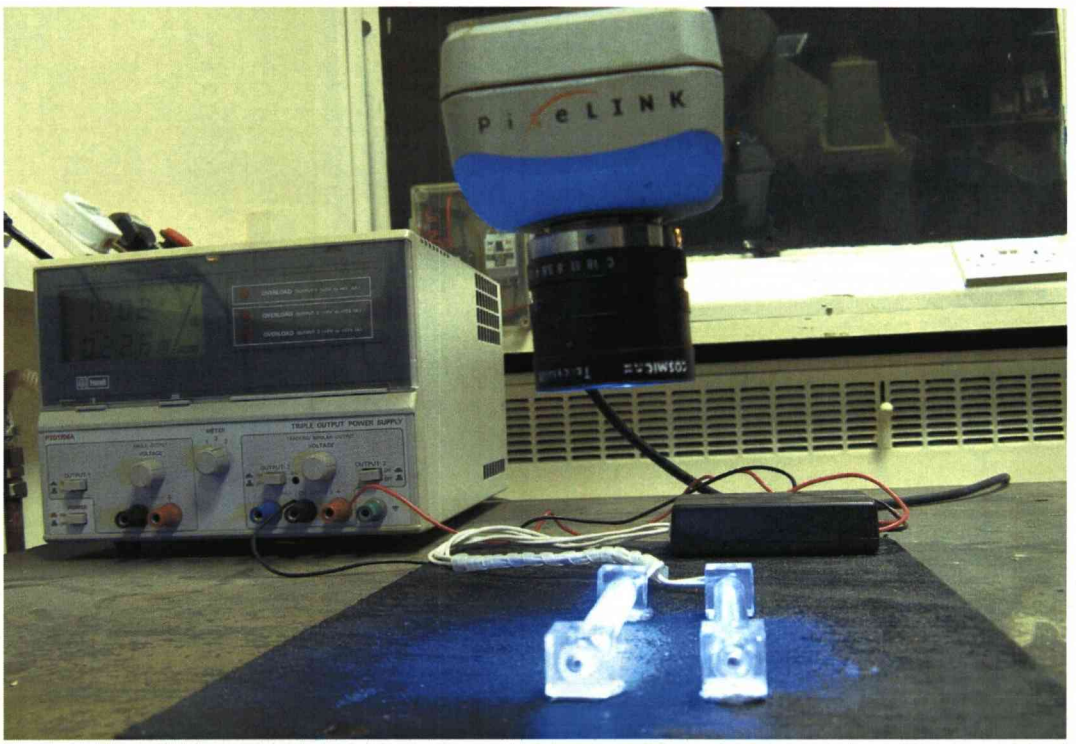

Figure 5.3: Target and camera, close-up, during a test.

5.1.3 Camera Ethernet Link

The task of communicating the parameters, determined by the image processing software on the camera subsystem computer, was carried out by a standard dedicated

100Base-T ethernet link. Each machine was fitted with a network interface card and assigned its own fixed IP address. These were linked with a crossover cable to eliminate possible bottlenecks, that may be caused by passing the data through a switch, in order to obtain as close to the theoretical maximum data rate of 100Mbps. Despite employing a dedicated link the image processing software and network link was very susceptible to system load on the computer and so all non-essential processes were terminated before the camera system was activated.

Server

On the controller computer, a soft real-time process was initiated with LXRT, see Section 2.4.4, which opens a port (700) on the IP address as well as acquiring the address of the shared memory area and its protective semaphore. Port 700 is then bound to a listening process which runs in a soft real-time loop continuously, within the Linux user space. When data is transmitted to the port the conditional statement within the loop executes and processes the data that has been received. Pseudo-code for this process would resemble the following:

- Make the process hard real-time to increase priority.
- Copy the data into a temporary store.
- rt_get_cpu_time_ns() - Record the time at which that data was transmitted.
- rt_sem_wait() - Test the shared memory semaphore, wait for access to the shared memory area (SHM).
- Lock the shared memory area to prevent concurrent access.
- Update the current error value with the new error value.
- Update the current error time value with the latest time value.
- Signal the SHM semaphore to relinquish access to the shared memory area.

- Return the process to soft real-time.
- Log the values to a logfile, when possible, to record performance for later analysis.
- Loop and keep testing for more data.

The logfile creation is done in soft real-time as it is not mission critical. Due to the differential rates of the control processes, the soft real-time logging can be preempted by a higher priority process, such as the one that wishes to use the data that was just sampled. The server process cannot be executed until the ISR module has been inserted into the kernel, as this initiates the shared memory areas and semaphores used by the server.

Client

The client side of the camera data link is embedded in the image processing software. On execution of the vision software, it attempts to connect to port 700 of the server machine. If it does not find the server present it alerts the user but continues in order to allow setup of camera parameters. If the server is present the client connects and awaits instruction. When selected, the processing software combines the image processing, pattern recognition and data transmission processes and executes each with every frame. Figure 5.4 shows an outline of the operations of the XP computer and the communications interfaces employed. The IEEE 1394b link transports the data from the camera to the computer system, which in turn is passed to the Linux

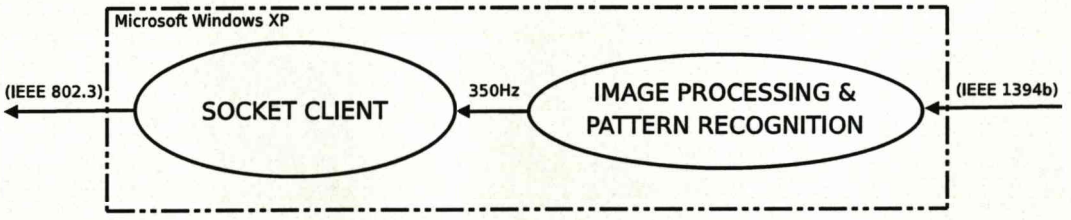

Figure 5.4: Camera subsystem operation.

control computer (server) via the IEEE 802.3 connection in a unidirectional manner.

5.1.4 Inter-process Communications

Computer systems use Inter-Process Communications (IPC) to pass data between any number of programmes, between programmes and the OS, sometimes between the programmes and hardware. IPC is a concept, executed in several different ways depending on what is required of the communication and what constraints are applied. Two methods are used to pass data between the processes of the system described by this thesis. Their places in the overall system are described in the overview chapter, Section 1.3, but they will be considered in detail here.

ISR to Control Task

The control task is passed updated joint data through an RTAI MBX structure, this is a mailbox which is analogous to a locked letter box. Messages, defined in size at the application level, are posted to a mailbox and retrieved from a mailbox. The mailbox service of RTAI is extremely flexible, allowing messages of custom and variable size. It also allows multiple senders and receivers on the same mailbox, with access order depending on the priority of the receiver task. The service also allows the sending of "oversized" messages where the unsent portion is stored until the receiver has retrieved the first section and stored it. Mailboxes can be used both in kernel modules and user space tasks, allowing user space tasks to communicate directly to realtime kernel modules instead of having to develop contrived data structures to achieve the same aim.

When a full packet of data has been decoded by the ISR, the joint angle bytes are combined into the 16 bit words they represent. These are then posted to the control task as a set of six joint angles, using RTAI's overwriting-send function. This posts a message of a designated size and overwrites any previous message, thus not blocking

the sending process which is the ISR. This ensures that any message retrieved by the control task will be the latest joint information. Initial tests were carried out using FIFOs, a faster form of message buffering that requires less processing, however the lack of protection and non-blocking features means that the transfer process was inelegant and harder to debug; mailboxes are fast enough for the task required currently.

The control task, the receiver, uses the rt_mbx_receive_if() function on the same mailbox. This function retrieves the message only if the entire message is available, if the message is not entirely available it returns immediately and informs the calling task that no bytes were received. This pairing process ensures that neither task will be blocked and will continue with the existing parameters until it is successful.

Control Task to ISR

For communication back to the ISR from the control task, the process of updating the joint positions with those chosen by the trajectory controller, another mailbox is used and the reverse priority is used. With this combination the control task updates the mailbox far more frequently than the ISR retrieves messages, with the overwriting function. As the control task runs at 1kHz and the ISR updates its position at around 36Hz the control task has the opportunity to correct its proposed trajectory before the ISR has chance to act upon it. Again, the ISR uses the rt_mbx_receive_if() function to retrieve whole messages only. Logging was carried out to determine how frequently updates from either mailbox were missed due to the use of the receive_if function - over the space of a ten minute test run no missed data events were recorded.

Commandline

It was often necessary to command the manipulator to take step changes in a single joint's position, for the purposes of assessing the joint controller performance as well as simply to move the manipulator to a more convenient position. This was carried out in a similar fashion to the main control mailboxes but was done using a one-shot "blocking send" from the commandline process. This process interprets parameters from the commandline and sends them to the ISR in the form of joint number followed by a displacement factor.

Shared Memory Area

In order to maintain a record of the latest camera error data, including its timestamp, a shared memory area was created inside the main kernel module. The shared memory area contains a data structure which in turn contains both the latest joint positions and their time of recording, as well as the camera data. This is then registered with the RTAI memory manager to allow access to it by LXRT userspace real-time processes. Along with the shared memory area, a protective semaphore was used to prevent two processes accessing the data simultaneously. The camera server process blocks until the protective semaphore is signalled as accessible by the control task, at which point the data is updated and the camera task releases the semaphore allowing the control task access again.

The control task, again, utilises a non-blocking function to access the shared memory area - this means the data into the control task may be "late", but the timestamping will allow correction when the next access occurs. The sections of code that access the shared memory area have been kept to an absolute minimum to ensure the least likelyhood of the shared memory area being in an inaccessible state when required by the camera. This is done by copying the latest data into a local structure before signalling the semaphore.

5.2 Operational Overview

Each interface, described in the previous section, was developed and extensively tested individually before being combined to create the overall system, Figure 5.5. The two-computer system was not the optimum solution, enforced by the requirement of a Windows^{TM} based image processing system. Ideally this would also have been executed on the RTAI Linux machine, eliminating the transportation delays involved in encapsulating the camera data in the TCP/IP transmission system and subsequent decoding at the Linux end. Chapter 4 discusses the reasons for not using the optimum solution. A further look at the Linux machine, Figure 5.6, shows more detailed description of the interaction of the processes outlined above, the camera data server, the control task and the ISR. Sharing data throughout so many tasks required tight control over which processes could read or write and when.

The vision control process has ultimate priority over all other non-interrupt driven tasks and as all other processes are implemented in non-blocking forms the visual control loop has free and guaranteed timely access to the sampled data and target data. Guaranteed, in terms of an RTOS, infers some variability of time of execution but within maximum limits that are infinitesimal in the overall system.

5.3 Control

In any process where a set point or target is to be reached by an imprecise plant or process, such as lightweight flexible or low-cost manipulator hardware, there are two methods of ensuring the system achieves its target despite its shortfalls in rigidity. The generally accepted method is to model the system [75, 76, 77], including its inaccuracies and compliance, in order to know exactly how it will react to inputs in any situation. For six DoF revolute manipulators, the equations of motion and dynamics are well established [27] and models developed have included details

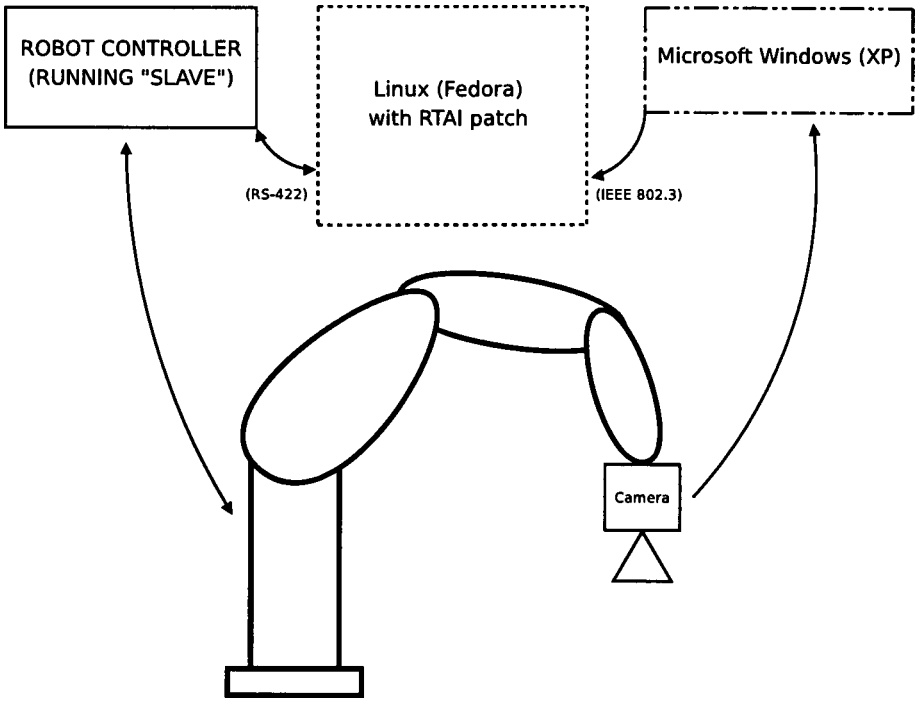

Figure 5.5: Operational outline of entire system.

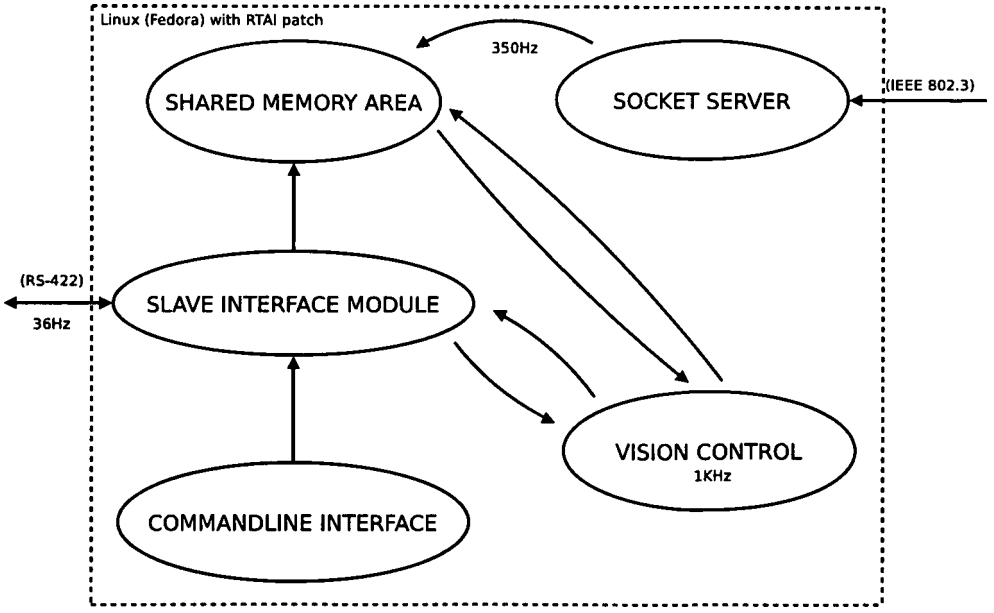

Figure 5.6: Linux machine operation outline.

as seemingly minor as drive motor armature inertia, as well as the gravitational and Coriolis forces generated by the link motion and position. Some work [29] combines both models and feedback from multiple sensor systems in order to give high performance of the controller system. The equations of motion, dynamics and the in-depth models are different for each style of manipulator and change with hardware modification and payload, as well as with time and wear to a lesser extent. Development of these models is time consuming and expensive, as are the added multi-channel control systems needed to assess and collate the information provided by the extra sensors that monitor acceleration and strain in the links.

The second method, proposed as a possible solution, is simple feedback control alone, at a very high sample rate. Slow sample-rate visual feedback is already in use in rigid systems and is sometimes used in flexible systems along with complex models and additional sensors as explained above. It is used give general target tracking and positioning while the model and accelerometers attempt to correct for the model error by predicting the next movement based on their model. The work of [29] shows good results in a two DoF testbed and in simulation on a higher order system, however is very complex and still uses modeling as well as corrective feedback.

In feedback control, sensor feedback is required to inform the robot controller that it has not reached the desired position or pose and needs to strive further to reach that target. A feedback control system can be represented in the form seen in Figure 5.7, where a setpoint value SP is given to the system, possibly in the form of a position or a speed, and the controller compares that value to the current process variable PV. An error is calculated and this is amplified and fed back into the controller in order to correct the PV with the new information. If feedback were not present, the system would be considered to be open-loop; errors in the process of target tracking would be present and uncorrected. Although there are many feedback schemes, feedback control is often achieved by proportional (P),

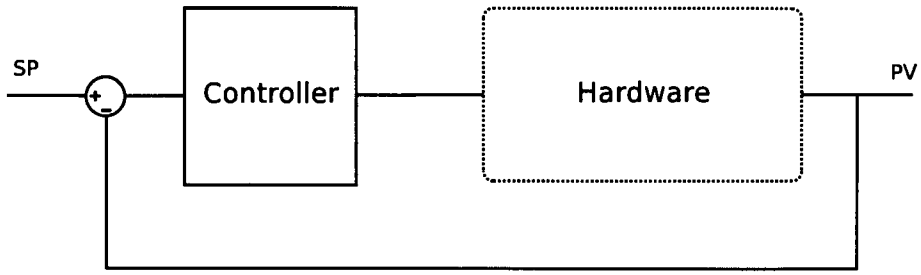

Figure 5.7: A basic feedback control system

proportional-derivative (PD) or proportional-integral-derivative (PID) in industrial applications, although other methods are becoming popular and the area is under constant development. Proportional control assesses the distance from the target and provides a corrective action to the controller output in order to ensure the controller reaches the position. Larger errors result in larger proportional correction. This is scaled by a gain, k_p, which needs to be tuned to the specific system.

Assuming a step input in the target, proportional control provides a high initial output due to the large discrepancy between the current position and the position indicated by the error signal, this is the dominant driving force in any such controller. This term diminishes as the target approaches and when the gain applied to the error signal is small enough in comparison with the resolution of the control system, there is always an offset from the target. This offset arises due to imperfections in the drive system such as friction in the gearing and backlash in the gears. In order to remove this error an integral term must also be used, although that may not be necessary if the system will settle to an acceptable error. The integral term and its gain, I and k_i, is a measure of the past error state of the system, in an analogue system the integral term is effective continuously from the initiation of the control circuitry, however in digital control the integral term can be generated by only integrating over a given number of samples. It will continue to attempt to remove an error from the system even when the system is stationary, the longer the system remains away from the target the harder the controller attempts to reach it. It will also, however, increase the likelyhood of overshoot, especially in a system with rapid control updates but

slow physical motion.

Finally the differential term and its gain, D and k_d, in a PD/PID controller compares the previous error with the current error in order to determine the rate of change of error, this term can be used to provide an anticipative action, looking at the current rate of change of error to estimate future error and act to improve it. The differential term is susceptible to noise and can cause instability at comparatively low gains. There are many variations on these PID control methods, the noted variations are parallel and series. In parallel PID the individual terms are calculated separately to each other whereas in a series implementation the terms are reliant upon each other in the calculation process and so interact. The parallel method is technically more effective and tunable due to the lack of interaction, but is more complex to implement in analogue systems - in digital control the differences are minor and the parallel implementation is usually chosen.

When considering long sample times, it is clear where large errors can accumulate between samples, leading to estimations of the next ideal location being far from accurate when finally applied to the motor or drive system. It is accepted that to control any system, the sample time T should be at most 10% of any time constant of the process being controlled,

$$T \leq 0.1T_p \tag{5.1}$$

This suggests that, should a manipulator oscillate at a rate of 10Hz, a sample rate of 100Hz would be required to accurately control the manipulator with feedback control. The higher the sampling rate, the better the control system is capable of following and correcting the system, with an infinitely small sample time (continuous control) being optimal. Higher sample rates, however, mean shorter sample times and, in digital control, less time to perform corrective calculations, requiring faster and more expensive sensor systems. Modern computer systems have progressed to a point where this is not the limiting factor in robot control, and camera systems that are a capable of high speed video are equally reducing in price to an affordable posi-

tion. The computer system employed in this research has the hardware specification shown in Table 5.1.

Table 5.1: Control computer hardware specification.

Item	*Specification*
CPU	AMD Athlon XP 1.83GHz, single core 333MHz FSB 128Kb L1 Cache, 512Kb L2 Cache
RAM	512Mb PC2700 (333MHz)
HDD	Maxtor 80GB SATA 1

During testing of the system, the time taken for execution of the of the full forward and inverse kinematics of the manipulator and PID routine, per cycle, was $14\mu s$ consistently despite a loaded system, this rate allows sampling at around 70kHz from a low-specification machine. In turn that would allow, assuming the sensor system were capable, good feedback control of a system with oscillatory response up to a frequency of around 7kHz, however the ultimate limitation on the control system in this case is the speed at which the hardware controller can update the joint positions. In the SLAVE system, as mentioned, the update rate is 36Hz meaning the hardware itself is only capable of satisfactorily controlling oscillations with a frequency of 3.6Hz or less. Despite this, results showing control can be achieved at these rates would be considered scaleable to a manipulator with higher hardware update rates - the PUMA control system is based on old hardware.

In order to test the basic response of the system created, a simple PID control system was created for the linear position of the camera in the x-z plane, working at 1kHz. The software implementation is described by Figure 5.8, as this shows, the system employed uses a more complex structure.

Due to the fact that camera error is a relative error, not known precisely in the world reference frame, in order to adjust the position of the arm to reach the correct location the software needs to know the following values:

- The current estimated position of the end effector.
- The actual measured error provided by the camera at the end effector.

If the system were a position based visual servoing system the camera would know precisely where the end effector was in 3D space and would not need to use the estimated position in order to gain a reference for feedback.

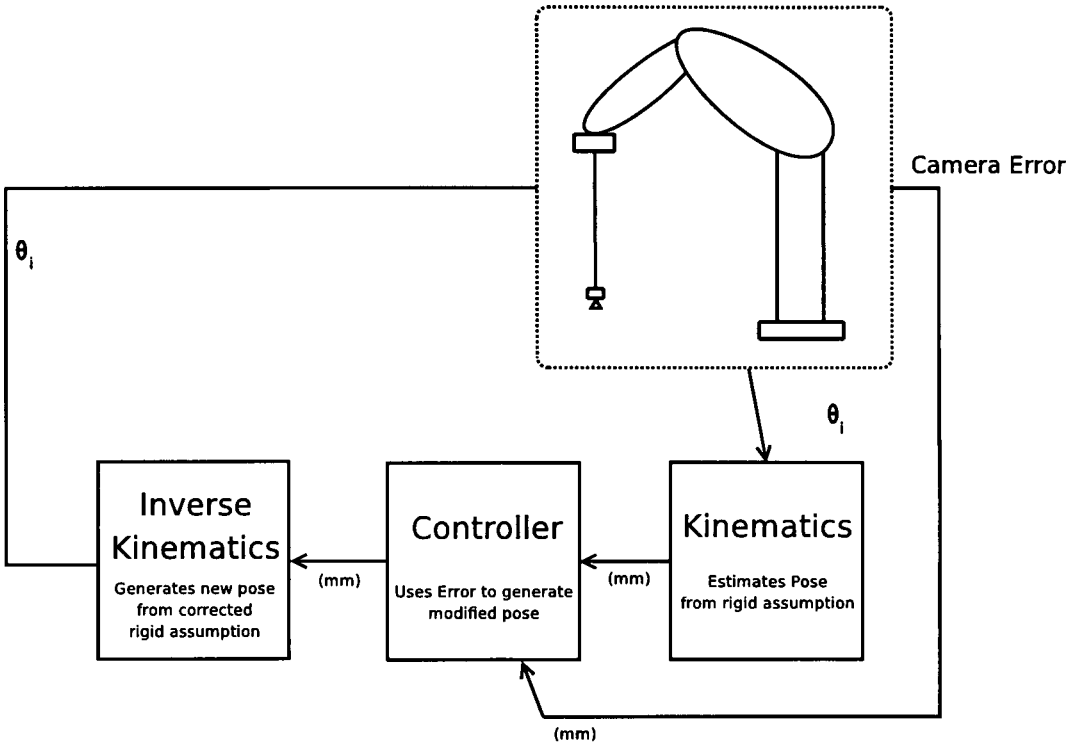

Figure 5.8: Overview of the controller implementation.

Using the three DoF kinematics, the position and pose as estimated by the kinematics can be represented by

$$E = \begin{bmatrix} x_e & y_e & z_e & \theta_e \end{bmatrix} \tag{5.2}$$

where x_e, y_e and z_e are the estimated coordinated in mm and θ_e is the estimated pose in degrees. In this research the camera only provided feedback for one cartesian axis, x mm. Therefore the camera correction data can be viewed as C, the error

from the target

$$C = \begin{bmatrix} x_c & y_c & z_c & \theta_c \end{bmatrix} \tag{5.3}$$

although the y, z and θ terms are all zero as this information was not extracted from the camera images, in the research carried out for this thesis, leaving

$$C = \begin{bmatrix} x_c & 0 & 0 & 0 \end{bmatrix} \tag{5.4}$$

In a single timescale control system, the camera feedback is observed and a PID routine described earlier (with gains k_p, k_i and k_d) calculates the level of controller output CO that is required based on C at a given camera frame z and for n previous frames,

$$CO_z = k_p C_z + k_i \sum_{z-n}^{z} C_z + k_d(C_z - C_{z-1}) \tag{5.5}$$

When this is combined with the estimation provided by the kinematics algorithm, the commanded position P_c for the next frame takes the form

$$P_c = E + (k_p C_z + k_i \sum_{z-n}^{z} C_z + k_d(C_z - C_{z-1})) \tag{5.6}$$

It is clear that the controller output CO and therefore the commanded position P_c is dependant on frames being recorded at equal intervals. This is the reason a real-time operating system was employed.

5.3.1 Tuning

In order for the controller to actuate a mechanism in a controlled and stable fashion the controller must be tuned to match the parameters of the plant. The tuning values can be gained through experimentation or through modeling and simulation of the mechanism being controlled, the latter is the accepted norm. Due to the constraint of not being able to model a manipulator with unknown, variable flexibility and/or backlash and payload, the solution is a self-tuning controller, however in order

to work towards development of a self-tuning system, investigations must first be done on the plant. These investigations are needed in order to determine critical information such as limits on joint drive current, imposed by external controller hardware.

5.3.2 Implementation

The following information details how the software and and hardware developed and investigated in Chapters 2, 3 and 4 were brought together and implemented as an operating visual servoing system in order to test the response of the system with high speed visual servo control. Figure 5.9 shows the hardware assembled and in the process of initial testing.

Per-Joint Controller

The SLAVE interface of the PUMA accepts only joint position information, at a fixed rate, therefore the controlling software must provide these and determine the correct rate of joint acceleration, the step size and rate of change of step size between the successive packets of information. The manipulator hardware will accept *any* position information be it out of range or too distant from the current reported position. In the event of an out-of-range position request, the control hardware will simply over-extend the joint until the limit switch in the hardware is tripped, at which point all communications will cease. In the case of excessive accelerations, large angle changes in a short time, the motor controllers will attempt to reach that position but will be tripped by their current limiting circuit which prevents application of excessive torque/current in the motor. Again, this triggers a hardware error and terminates communication with the control computer as a failsafe.

In order to prevent either of the two situations above, initially the ISR routine included a per-joint control algorithm intended to optimise the speed of response

Figure 5.9: Robot arm, flexible link, camera and target in action.

of each joint without overshoot or over-current cut-out. ZN tuning methods were impossible due to the rigid nature of the links - increases in proportional gain did not produce oscillations. Although successful in this task the joint controllers interfered with the main visual control loop and created both disparity and instability. The effect of reducing the initial acceleration of each joint caused the kinematic

assumptions to become incorrect as each joint suffered lag, taking the robot away from its instructed pose. The visual control loop attempted to correct this but the combined controllers caused oscillations and unacceptably long response times. As the controllers were tuned to reduce the oscillations it became apparent that the most effective system was to simply eliminate the per-joint control and pass joint angles directly from the vision loop to the joints. This method produces the most rapid propagation of joint angles from the visual controller and can produce over-current problems in a situation where the target is lost. In order to prevent this situation, the image processing software was modified to return an error of 0 should the target be lost - this puts the manipulator into a holding state instead of allowing it to produce large, unexpected, translational motions.

5.3.3 Visual Control

As described earlier in Figure 5.8, the visual control loop incorporates feedback from the camera subsystem and the current estimated pose from the joint encoders and kinematics. Both feedback signals are at different sample rates, the camera operated at approximately ten times the pose estimation feedback rate, with the overall controller updating at three times the rate of the camera sampling, as shown in Figure 5.10. From this point on the term control loop will refer to the

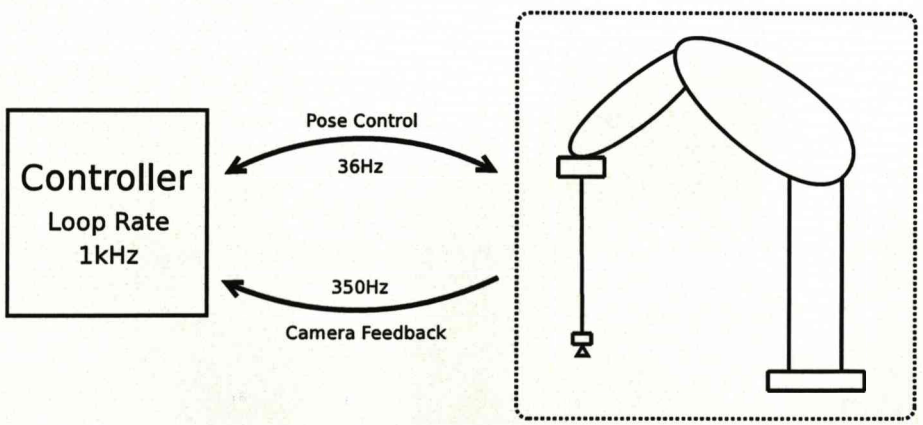

Figure 5.10: Important control sample rates.

1kHz calculation loop which carries out the calculations of kinematics and inverse kinematics and the calculation of the next positional output, the camera system is the 350Hz visual data feedback subsystem and the manipulator system is the 36Hz hardware controller used to actuate the manipulators motors and read the joint encoder system.

Because the overall control loop executes far faster than the data is sampled, it can take make use of the latest changes in feedback from the camera system before outputting new joint angles to the hardware controller. The controller takes one control period to transfer the position to the joints themselves, introducing a minimum 28ms lag from the latest camera error data.

5.3.4 Two Timescale Nature

As explained by Bascetta [29] and in section 6.4, flexible manipulators exhibit a two timescale nature. This is the a dual response to disturbance whereby there is overall slow movement towards the target position, but with a higher frequency oscillatory response due to vibration and resonance superimposed onto the slow movement. It can be shown that, without oscillatory cancellation, the average position of the manipulator will follow a relatively smooth path despite violent tip deflections due to vibration. This smooth average path can be considered to be the rigid response, while the rapid, flexible, end effector motion is oscillatory around that smooth response. The manipulator has both a rigid response and a flexible response, separating the control problem into a two timescale problem, fast and slow. The fast control problem is that of removing the oscillatory motion in the link that is superimposed onto the average smooth motion. To do this the fast controller must know the current assumed-average position calculated by the slow controller, as well as utilising the high frame rate data directly from the camera.

For this task the 1kHz overall control loop was separated into a two timescale system where one sample in a given number is compared to the latest manipulator positional information, creating a positional control loop that can operate at a range of frequencies as selected at runtime. The fast control system works simultaneously with the slow loop, analysing all of the data between and including that used by the slow loop. Instead of comparing it to the overall positional displacement from the target, it is compared to the latest value recorded by the slow control loop. This aims to assess the level of oscillation against the current error from assumed-rigid position of the manipulator without affecting the overall position greatly. This separation of high and low speed controllers allows the gains of the high speed controller to have a lower effect on the overall position control while allowing fast and firm response to the oscillations, Figure 5.11.

Recalling equation 5.5, it can now be seen that the controller has been split into two PID controllers, slow CO^s and fast CO^f, hence CO^s takes its original form but operates on less frequent data updates at frame time t, here called C_t^s. The slow and fast controllers also employed their own gains for each term, seperating these into ${}^s k_p$, ${}^s k_i$, ${}^s k_d$, ${}^f k_p$, ${}^f k_i$ and ${}^f k_d$.

$$CO_t^s = {}^s k_p C_t^s + {}^s k_i \sum_{t-n}^{t} C_t^s + {}^s k_d (C_t^s - C_{t-1}^s) \tag{5.7}$$

and the fast controller

$$CO_z^f = {}^f k_p C_z^f + {}^f k_i \sum_{z-n}^{z} C_z^f + {}^f k_d (C_z^f - C_{z-1}^f) \tag{5.8}$$

but CO^f uses the last C recorded by the CO^s routine as the reference by which it determines its own error input C_z^f

$$C_z^f = (C_z - C_t^s) \tag{5.9}$$

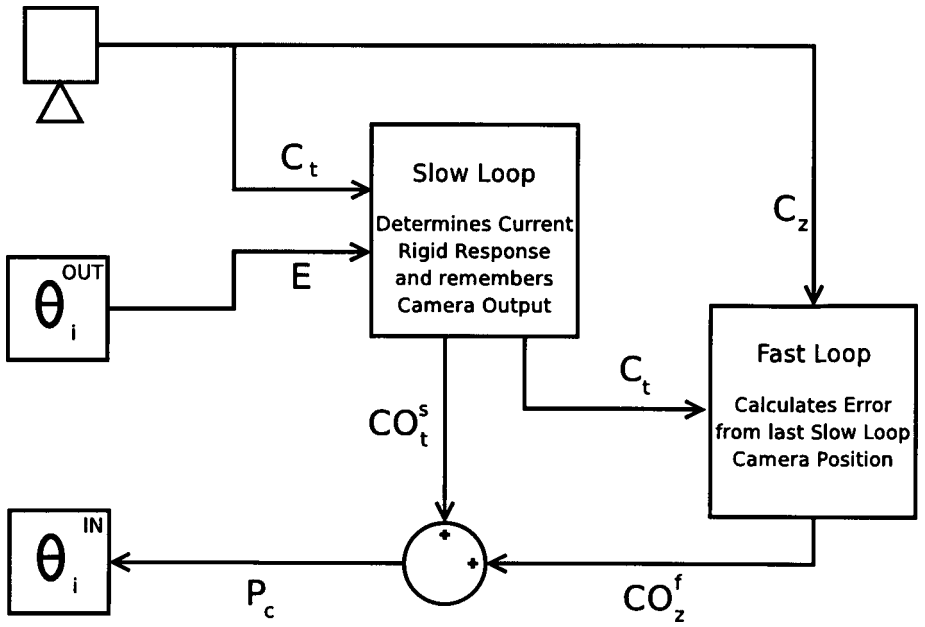

Figure 5.11: Structure of the two timescale controller.

therefore the overall controller output CO is given by

$$CO = CO_t^s + CO_z^f \tag{5.10}$$

and the next output commanded by the two timescale controller is given by

$$P_c = E + (CO_t^s + CO_z^f) \tag{5.11}$$

Selection of the fast controller frequency is done by selecting a rate which is at least as high as the fastest feedback sampling rate. If the two feedback signals are collected and stored simultaneously this minimum rate will utilise all data with each execution and further repetitions are wasteful. However, if the signals are asynchronous, multiple calculations between the fastest sampling will allow more timely corrections utilising the latest data from all signals despite some level of latency. For this reason the fast feedback loop frequency remained at 1kHz, the frequency of the overall controller loop. Due to the available excess of processing

power this was not problematic and left sufficient processing power to run a heavily loaded Linux desktop system simultaneously.

The slow controller rate is more complicated to select, a low freqency system will give large latency in initial movement after an input change, however a faster system will reduce this but may follow the shape of the oscillations just as the fast controller does. A lower frequency slow controller will allow large gains to be implemented to allow more rapid manipulator movement, whereas with higher frequency input those high gains will cause vibration in the position control and instability, the ideal frequency is a compromise, dependant on oscillation rate and magnitude. From experimentation, the ideal slow controller frequency is approximately twice the natural frequency of the system being controlled.

In the slow loop controller the proportional gain is of primary concern, forcing the manipulator towards the target position, integral gain can be used to settle the manipulator at zero however, of more use, a negative differential action can be used to reduce the sharpness of response which tends to cause oscillations in flexible manipulators, and allows the proportional gain to be increased. Increasing the proportional gain reduces the offset from target and decreases the settle time, while the differential gain tames the initial motion, without having as great an attenuation when closer to the target.

The fast loop controller gain requirements are somewhat different to those of the slow loop, the overall intention of the fast loop is to reduce error from average without setting up further oscillations as a consequence of forcing rapid motion. Analogous to how it is possible for a human to move a beaker of liquids rapidly without spilling them, the fast loop acts primarily with differential gain; As the end effector oscillates, it travels with and against the overall direction of motion of the manipulator, in approximately sinusoidal form, with peak speed occurring as it passes the average position calculated by the slow controller. The strain energy in the beam, due to differential motion of the two ends of the flexible link, causes the

oscillations. In order to reduce this strain energy, the differential term attempts to maintain the tip velocity at the same magnitude and direction as the slow controller. A small quantity of proportional gain may be required to remove small oscillations when near the overall target.

Figure 5.12 shows the effect of the two timescale control system on the manipulator position, describing the overall manipulator motion, left in this instance, towards a target. Mid-oscillation, the end effector of the manipulator is likely to be, at least part of the time, moving opposite to that of the overall manipulator motion - in the diagram this is represented as towards the right. Finally the fast control loop direction of operation is also to the right to minimise the end effector acceleration. The fast control loop attempts to force the manipulator to accelerate and decelerate with the end effector in order to damp out the oscillations. This is because moving in the opposing direction to the end effector only stores more energy in the flexible beam for future oscillations. This has to be overlaid over the general motion of the end effector and is a compromise between oscillations and overall motion, if the oscillation control is too high a priority, the overall motion would cease. Humans carry out this process when carrying a glass of water, allowing the glass to move with the liquid to prevent spills due to oscillations, whilst still moving the glass towards the coaster.

Both the parameters of the slow and fast controllers depend on robot characteristics such as joint drive capabilities, link elasticity, payload level and position. Due to this the parameters would need tuning in-process if any of these change, although in a constant repetitive process these could be fixed at optimal. The tuning of these, ultimately, could be done by analysis of the overall camera feedback either in an on-the-fly method or as part of off-line calibration. An investigation of the parameters is included in this research in Chapter 6. Functionality of this simple two timescale system proves that it is possible to utilise only high speed camera feedback in image based visual servoing as a control method for a flexible link manipulator. This

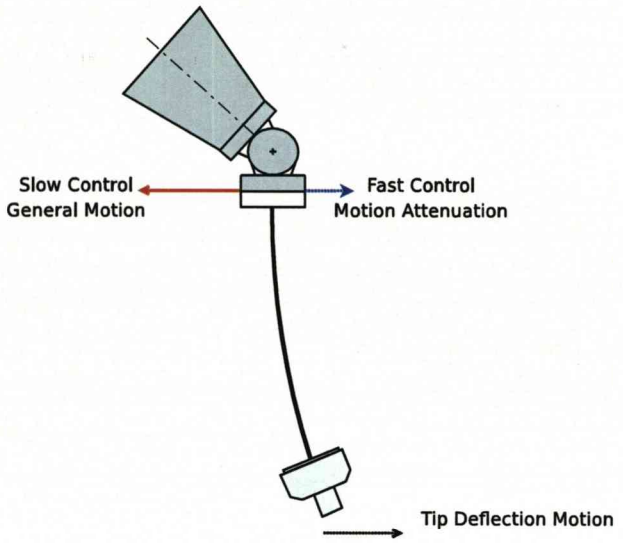

Figure 5.12: Strain energy reduction by manipulator speed adjustment.

should, given ample increases in the hardware interface rate and greater information extraction from the image, be capable of position and vibration control of a fully flexible manipulator created with lower quality drive mechanisms and links. Single camera feedback reduces the sensor integration requirements of the system, making it inherently less expensive.

5.4 Summary

This chapter has presented the challenges inherent in interfacing a robot manipulator and a remote camera system with a central control computer system. It discussed the specific interface details such as the dedicated ethernet link to the remote camera that was implemented due to system compatibility issues. The driver that was created to interface with the robot hardware through SLAVE was investigated and outlined. Further details of the methods of communication between each of these processes and drivers was documented.

An overview of the entire system function was then given, showing the interface rates of each component, and diagrammatic representations of the overall control

systems employed. Details of the control methods used were explained, single and two timescale control were considered and reasons for using them were given. A review of other control methods and other research in the field of intelligent control systems is given in Chapter 8. Due to the modular nature of the software created, the application of these techniques in future work would take little work and has the potential for much improved system response.

Chapter 6

Experimental Results

As has been shown in the previous chapters, a flexible test system has been developed in a modular fashion whereby various control techniques can be implemented within the high speed control loop section of real-time code running on an RTAI-Linux based computer. A large amount of information has been made available to the control system including timestamping of data, both inbound and outbound, providing for more complex systems to take full advantage of noting the lag times between each arriving (or leaving) variable in calculating either predictive or responsive control results without being affected by any transport delays. This is unusual for a control system, where data normally is simply assumed to have arrived at a set time, introducing timestamped data allows more flexibility in the sensor system design, although greater complexity in the controller design.

As has also been shown, a simple two timescale PID feedback system was developed in order to test the hypothesis that a high speed camera alone could be used in image based visual servoing for end effector trajectory generation and oscillation reduction on a flexible or degraded performance manipulator. The following results show that the manipulator was successfully visually servoed using this control method, identifies important features of the control system and highlights possibili-

ties for future improvements to the system based on the results returned, which will be discussed in further detail in the following chapter.

6.1 Testing Procedure

In order to test the effectiveness of the camera feedback controller to remove both steady state errors and oscillatory motion of the degraded manipulator, a simple step input test was devised. Although, in the proposed target application of robotic welding, step inputs are not common - usually weld placement follows a smooth and connected line trajectory - the step input is representative of motion towards an unknown target within a general area at the beginning of a welding cycle.

It is proposed that the closed loop vision system would only become active when the manipulator is near the target, when the line tracking algorithm would actually have a target to track, in order to place the manipulator correctly and smoothly despite misalignment of either workpiece or robot, and to eliminate oscillation as rapidly as possible for weld commencement. The time taken to settle on the target is of as much importance as the correct placement of the end effector as this will partially determine the rate of throughput of the welding system - large delays in manipulator targetting mean reduced overall rate and lost productivity.

The simulated test environment featured the cold-cathode target system, representative of the weld bead target, and the flexible link designed earlier. A 200 pixel step change was selected so as not to exceed the high speed cameras windowed area with the predicted oscillation size. The camera height above the target was fixed within the kinematic target calculations as the data from the camera system was not sufficient to determine range, although in the case of robotic welding, the distance from target can be estimated online by the arc voltage, which is proportional to arc length. This could be used in the control process as feedback to self-correct the end effector height.

Initial testing used two cold cathode tubes a fixed distance apart wired to light in opposition with a switch. Unfortunately, cold cathod tubes have an initial power-on delay, combining this with the time delay introduced by the mechanical switch system the fast controller was capable of identifying this period with no target. Instead, a software solution was devised whereby the active window within the overall image was shifted by a fixed pixel quantity. This is effectively identical to and indistinguishable from an instantaneous physical change of location of the active cathode tube, without any hardware changing location. The procedure for testing was as follows:

- The manipulator was loaded with a selected mass, depending on the test selected.
- The controller gains set to zeros, by command-line interface. Zero controller gains prevent the manipulator from moving in any way, both for setup reasons and for safety.
- The camera server software on the Linux machine was initialised and awaited connection from the camera client.
- The arm was stabilised to remove any initial motion, then the camera client was initiated and connected to the server - the server acknowledges connection and awaits data.
- The target was aligned under the camera so that it was centre-frame, with zero error and the image processing algorithm was initiated, transmitting data to the server which initially discarded it due to the zero controller gains.
- The controller gains were then set to the required levels for the test.
- The position of the left hand side of the image processing window was then shifted by 200 pixels and the response to the target motion was recorded by the server on the Linux system as a continuous logfile.

- The reverse direction was also tested for each gain set.
- Subsequent tests on different gains, at the same end effector mass, were carried out when the oscillations had subsided and the target re-placed at the centre-frame location, before adjusting the controller gains and repeating the process.

The logfiles retain data from the camera server system, logging both real-time CPU timestamp in nanoseconds and the error recorded, at that time, by the camera system as in Table 6.1. These results were then fed into Scilab, a data manipulation and display package available from *www.scilab.org*, to confirm their overall appearance and ensure no faults had occurred in the logging system. An example of raw logfile output can be seen post-processing in Figure 6.1. In this log file the results are those of increasing proportional gain, to the point of oscillation. These results were then taken and separated into individual movements for comparison. Due to the sheer quantity of results, only selected results are included which demonstrate the effects explained in the text. The tests were carried out with two fixed masses in order to give an example of both high inertia payload and low frequency oscillations, as well as lower inertia payloads and higher frequency oscillations, to show the effect of these changes on the control system requirements.

Table 6.1: Example logfile data.

Timestamp (ns)	*Error*
812190597879	188
812192594415	188
812194842762	187
812196669355	186
812198612959	185
812200657318	184
812202619917	183
812204618709	183
812206625763	182

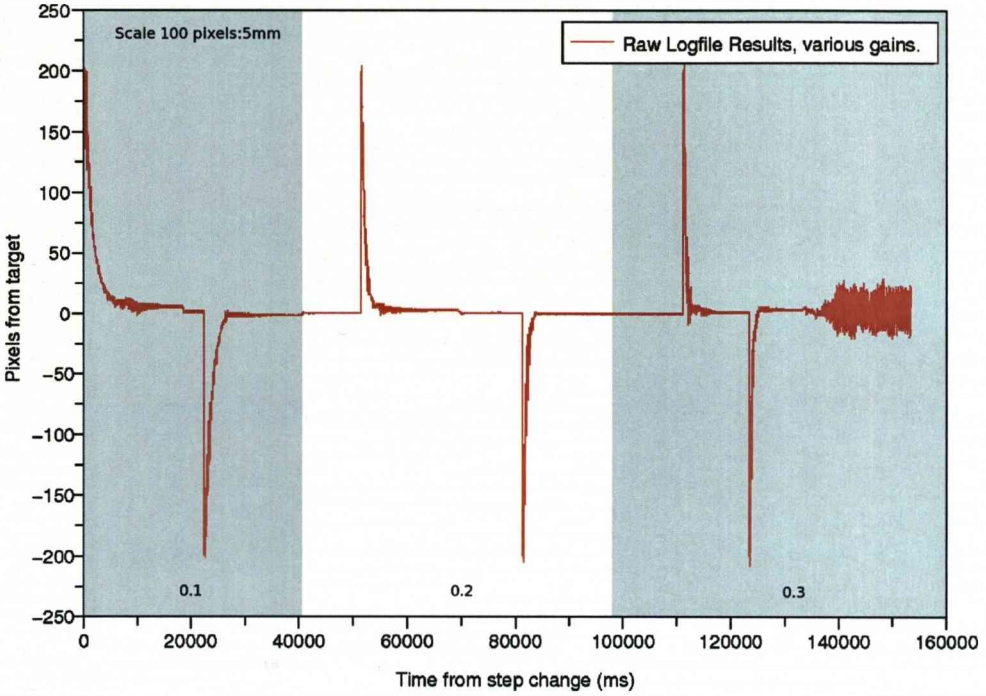

Figure 6.1: Raw logfile output pre-processing, 0.1, 0.2 and 0.3 proportional gain with 10Hz controller.

6.2 VAL Testing

As the system proposed is a closed loop system, it was considered necessary to first consider the open-loop system of the original manipulator. As the open loop VAL control has the dynamics of the original robot included in its trajectory planning and is not responding to a step-change input in the same manner as the visual controller will, it can set its acceleration and deceleration in order to move the end effector in a smooth and accurate fashion. It cannot, however, account for the flexible link deflection and subsequent oscillations, this will be demonstrated by the following results. Due to the symmetric nature of the results, in order to remain clear, only the positive movement results are presented.

Figure 6.2 shows the response of the VAL system, with 0.3kg end effector mass, at VAL speed 100. The dotted lines either side of 200 show a primary ± 10 pixel error

band and the solid lines show a secondary ±5 pixel error band. At this target range these represent $\pm 0.5mm$ and $\pm 0.25mm$ deviations from target, respectively. These bands allow comparison of system response across the differing controller systems, including a delineation time of convergence on the target. The time of convergence is determined to be the last point at which the output enters the required accuracy zone and crosses the target line without leaving the zone again.

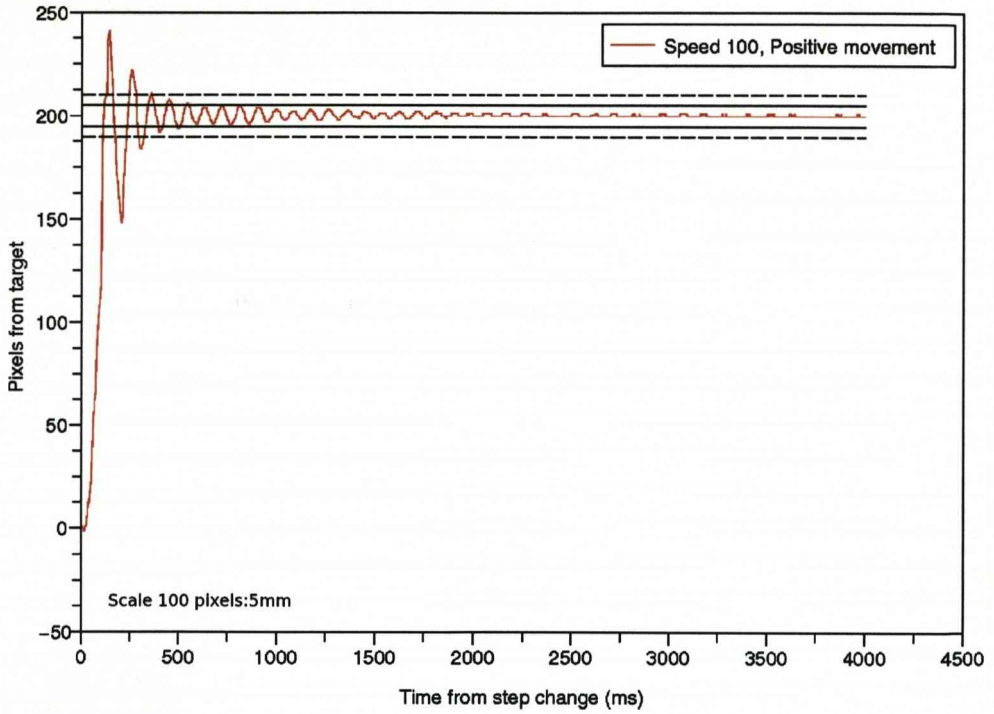

Figure 6.2: VAL control, speed 100, movement with 0.3kg end effector mass.

As can be seen from closer inspection of Figure 6.2, the system achieves primary convergence at 375ms and secondary convergence at 550ms with the 0.3kg load and a rapid rate of oscillation (11.1Hz), as noted in Chapter 3 Section 3.4, that slowly decays further to zero - ignoring the camera sample alias of one pixel, though this is not visible here. Testing of VAL speeds from 0.1 to 100 relay some interesting results, Table 6.2. With the introduction of a larger mass, 3.5kg, the manipulator can be seen, as expected under the mass-spring system assumption, to oscillate much

Table 6.2: VAL controlled primary (P) and secondary (S) convergence times.

VAL Speed	3.5kg P(ms)	3.5kg S(ms)	0.3kg P(ms)	0.3kg S(ms)
0.1	3750	3800	3660	3660
0.25	3750	3750	3750	3750
0.5	2750	2750	2950	2950
0.75	2050	4500	2000	2000
1	1800	4600	1700	1700
5	600	600	400	500
10	2300	4500	260	350
20	2150	4000	250	310
30	-	-	350	550
40	-	-	400	575
50	-	-	350	575
60	-	-	300	550
70	-	-	375	875
80	-	-	350	550
90	-	-	350	570
100	-	-	375	550

slower and with greater amplitude. This higher mass, especially at a position that was distant from the centre of rotation of Joint 5, lies outside the payload limit for the standard manipulator at VAL speed 100, therefore testing began at speed 10, Figure 6.3 and progressed until the system was unable cope with the forces involved. At this point, speed 30, the following problem occurs.

The acceleration and subsequent deceleration of the mass exceeds the holding torque of the motor in Joint 5, the torque is outside that controllable by the joint hardware controller and amplifier boards. This causes the the mass to swing away from the target position, suffering not only from the deflection of the flexible link but also causing the control system for that joint to strive to reach its correct position. This causes the link, eventually, to be oscillated in increasing magnitude until the joint control amplifiers register an over-current situation and remove power to the joint to prevent damage. On Joints 1, 2 or 3 this would cause the electromagnetic brakes to be activated but on Joint 4 and above this does not happen and the joint resists further motion simply by the highly geared nature of its drive system, as visible in Figure 6.4.

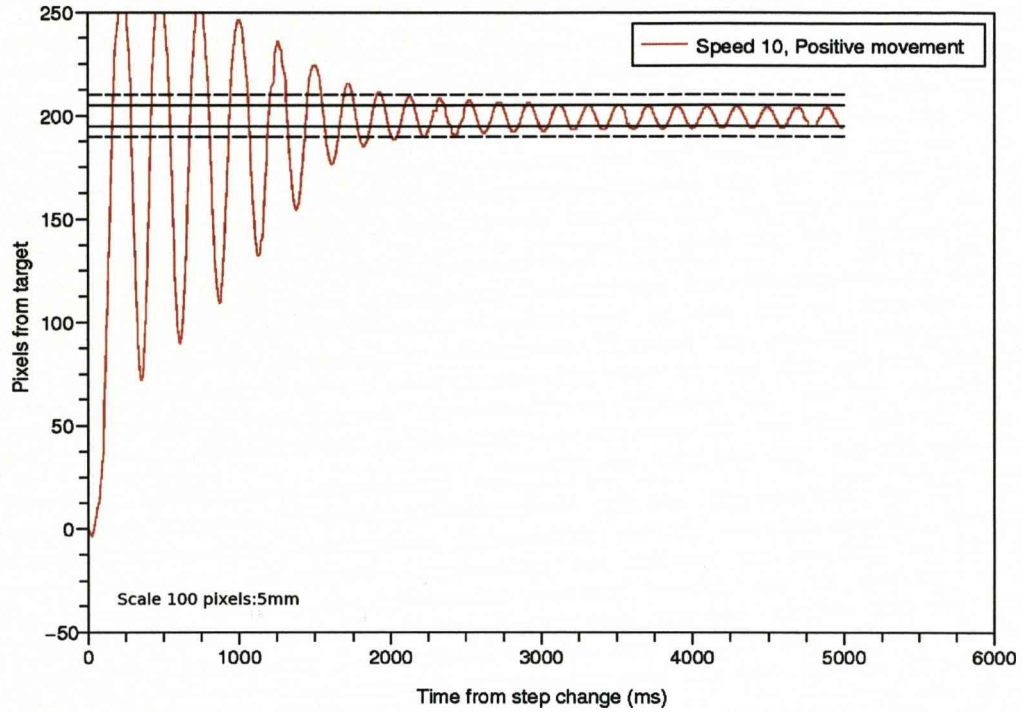

Figure 6.3: VAL control, speed 10, end effector movement with 3.5kg mass.

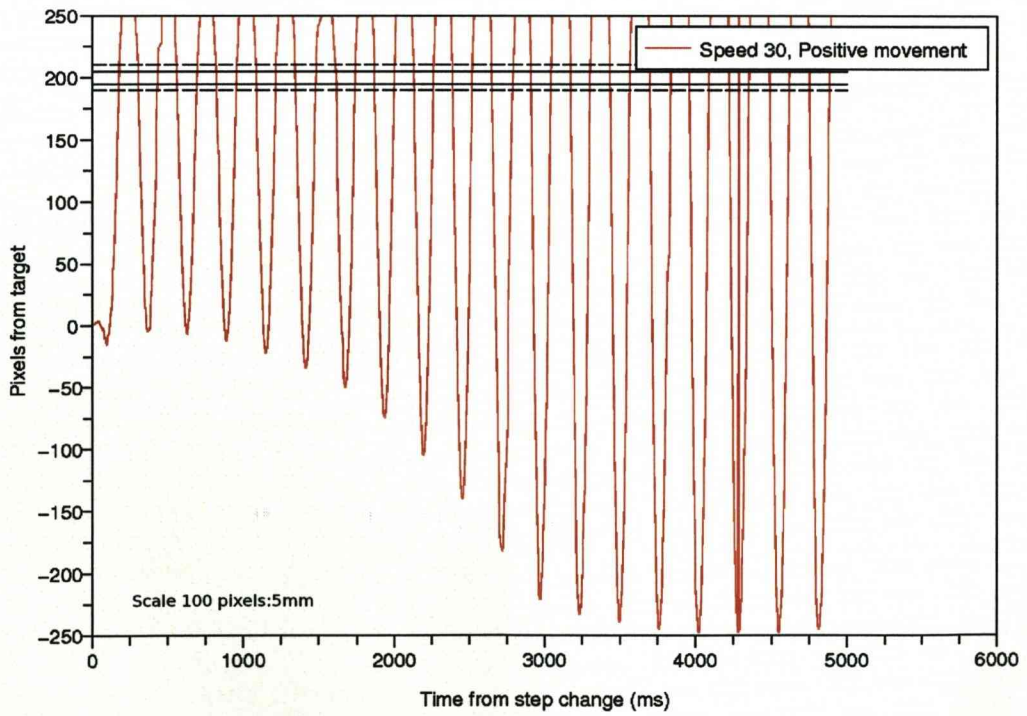

Figure 6.4: VAL control, speed 30, end effector movement with 3.5kg mass.

VAL speeds were tested to determine what the speed nominal numbers represent in relation to manipulator motion. A sequence of tests were carried out, moving the manipulator over a fixed, known, distance. The time taken for these movements to be completed was measured over a series of repeats and an average taken. The results of these tests can be seen in Figure 6.5.

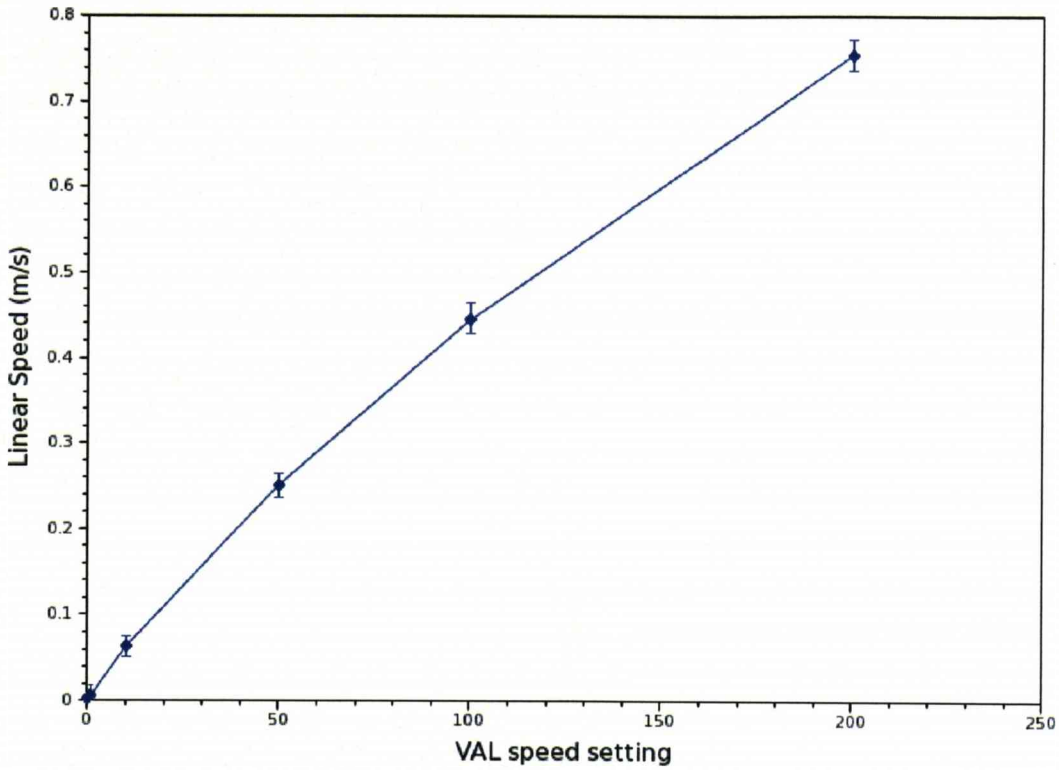

Figure 6.5: Manipulator end effector speeds in relation to VAL speed setting.

With a higher mass attached it can be seen that even at speed 10, 14.3% of the linear speed of the 0.3kg test, the manipulator only manages the primary convergence target at 2000ms and secondary convergence at around 5000ms, Figure 6.3. This is considerably longer than with low mass, and shows how manipulator models can become totally inaccurate with the introduction of a relatively small increase in mass, greatly affecting the performance.

6.3 Single Timescale Visual Feedback Testing

In order to prove the difference between single time-scale and two time-scale control, testing was performed with a PID visual servoing loop running at 100Hz, with both 0.3kg and 3.5kg masses. In these tests VAL is no longer responsible for the speed of response and is no longer part of the control system, speed commands are ignored. In order to investigate the effects of increasing proportional gain on the controller and to establish the limits of it, the two masses were attached and testing carried out as per Section 6.1. As expected, the results for the 0.3kg mass show little oscillatory motion, apart from that immediately after the initial movement, and settle to the secondary convergence margin within 1700ms. As the system does not cross the the target position, 0, the settle time is taken to be the point at which the system goes under the margin without oscillating back out again, Figure 6.6. After the proportional gain was raised to 0.4 the system became unstable, large oscillations caused rapid joint velocities to be demanded, the robot's hardware controller terminated communication. With the 3.5kg mass the system converges 400ms slower, Figure 6.7, due to the effect of the large oscillations past the target location causing the controller to reverse its general direction of motion temporarily, the convergence (both secondary and primary) are longer than with 0.3kg mass. This data shows the overall response times are similar to the speed 10 VAL open loop control, although after the point of primary convergence the visually guided system dramatically reduces oscillations in both situations. The primary convergence is 1800ms while secondary is achieved at 2500ms. The offset from the target is identical on both systems despite the maximum the proportional gain on the 3.5kg test being lower than that of the 0.3kg test. Increasing the the differential gain on this test, in order to slow the initial response and reduce the oscillatory action with the aim of shortening convergence time, caused instability due to the initial oscillations.

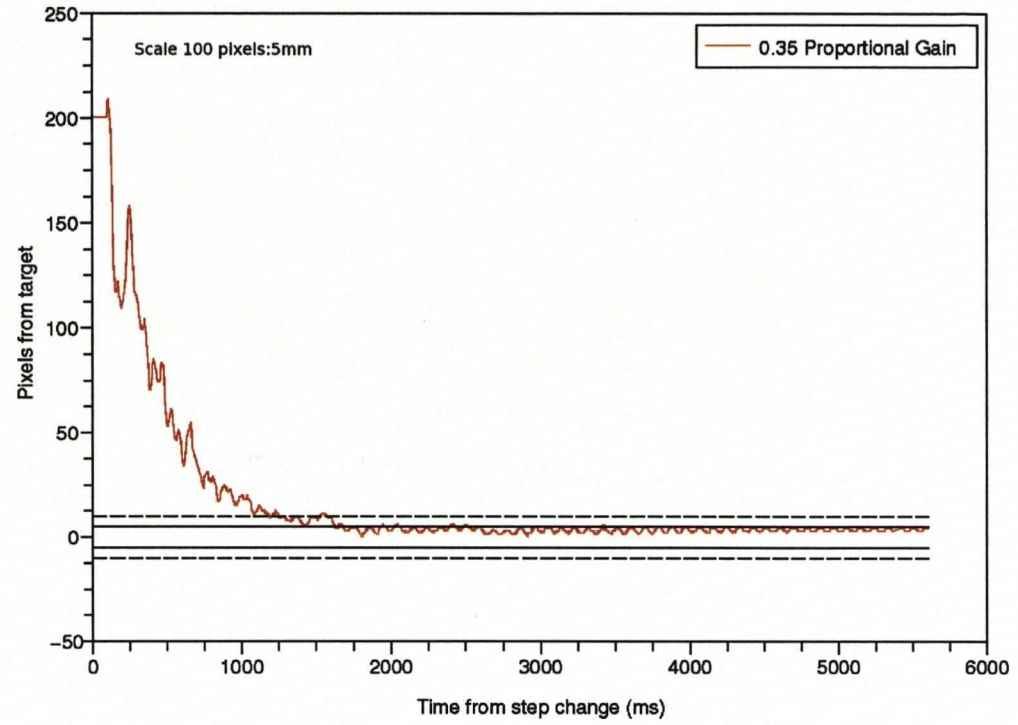

Figure 6.6: Single timescale prop. control, 100Hz, end effector movement with 0.3kg.

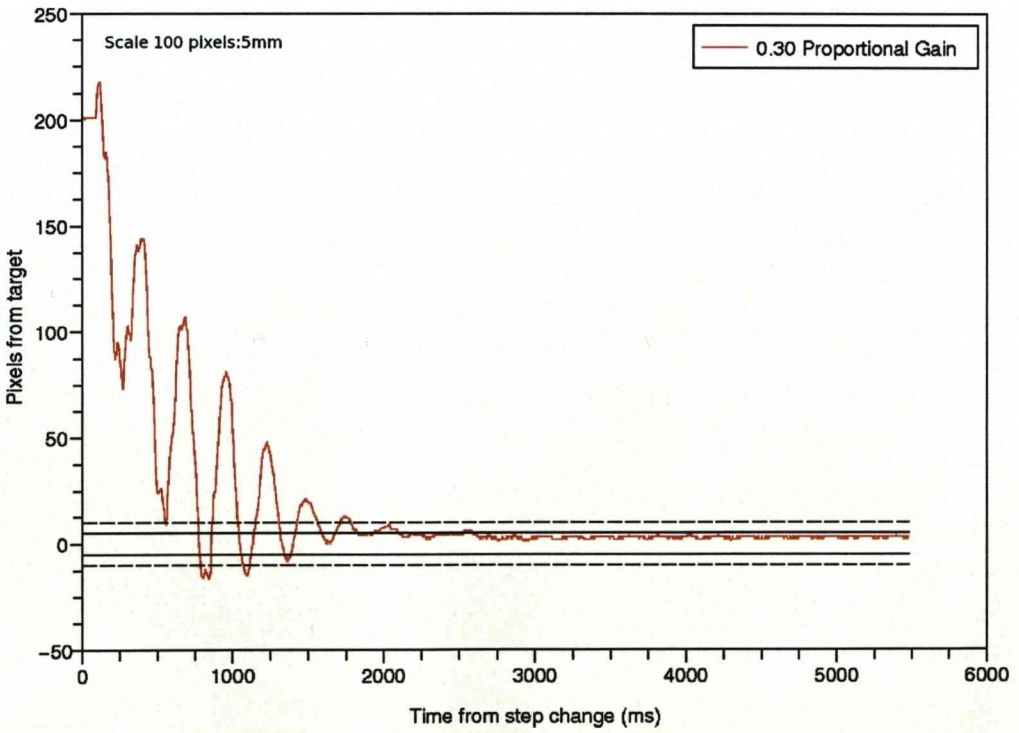

Figure 6.7: Single timescale prop. control, 100Hz, end effector movement with 3.5kg.

Features to note are the 80ms delay before manipulator motion commences, probably caused by a combination of transport delays in the system including the delay between the data being recorded by the camera (3ms) and being used in calculation (up to 10ms), up to 28ms delay before incorporation into the interrupt service routine memory, then a further 28ms before action by the motors with some IPC delays, although these do not fully account for the 80ms. The second feature of interest is the short positive swing before the negative motion of the main swing towards the target. This was determined to be due to the inertia of the mass, hence its lower amplitude in the 0.3kg test, and the rotational effect the bending moment has upon the tip of the flexible link causing the camera to point in the direction opposite to that of travel, this error is an inherent feature of end effector deflection and is present in all of the system responses.

Further testing was carried out at different rates of control, other than 100Hz, namely 5Hz (Figure 6.8), 10Hz (Figure 6.9) in order to determine how much and what nature of effect the controller frequency has upon the motion of the flexible link with the 3.5kg mass in single timescale control. At 5Hz and 0.1 proportional gain, the controller was unable to move the manipulator while close to the target which led to a large offset error after the step change, which took 6000ms to reach. With increasing proportional gain the system became less stable, although still operable, oscillating around the target position at the natural frequency of the link despite being physically restrained on several occasions. The primary settling time was reduced but the instability caused by the control period being similar to the natural frequency of the beam meant the narrower, secondary convergence target was not met. Increasing the gain further to 0.3 made the system even less able to maintain a steady position and eventually became totally unstable and diverged to failure after almost settling to primary convergence, after the 200 pixel step.

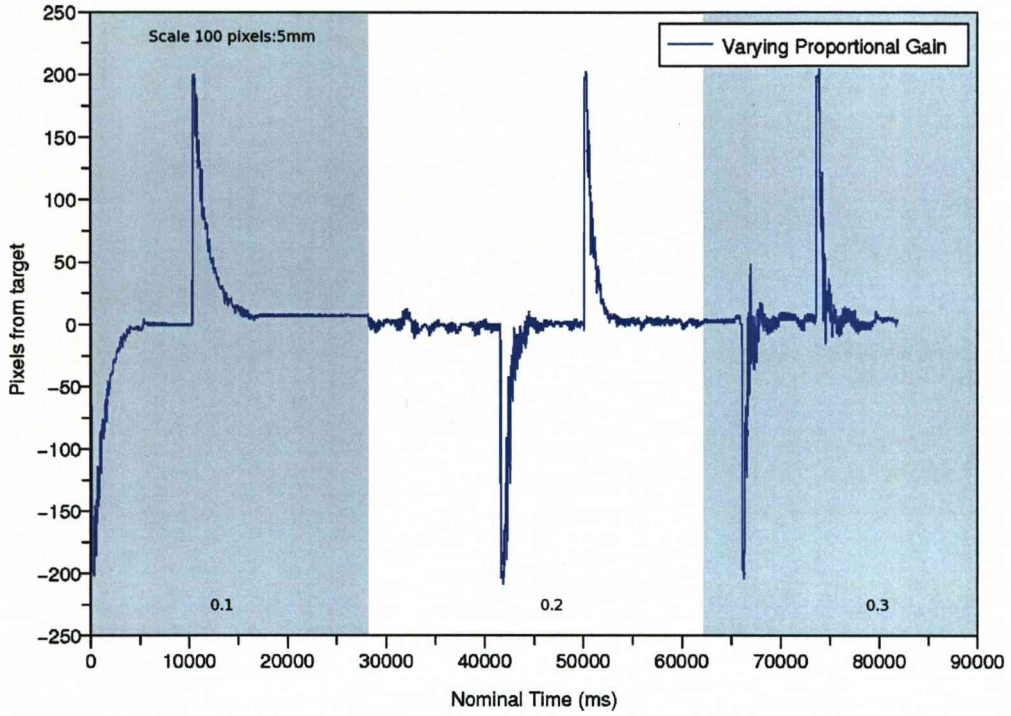

Figure 6.8: 5Hz Single Timescale controller at 0.1, 0.2 and 0.3 proportional gain.

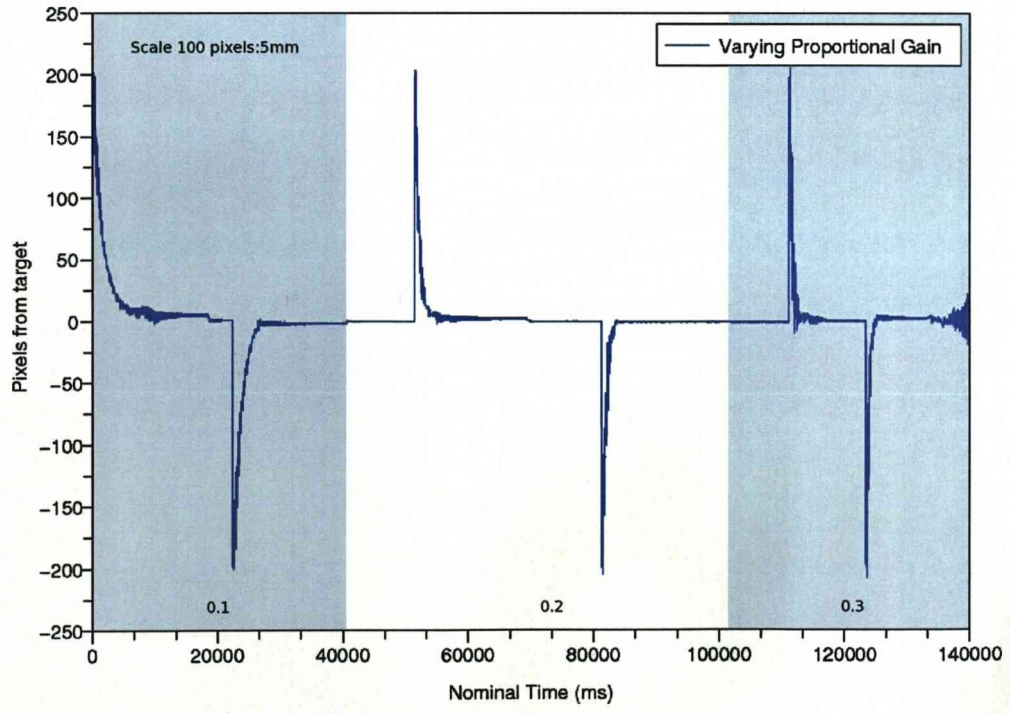

Figure 6.9: 10Hz Single Timescale controller at 0.1, 0.2 and 0.3 proportional gain.

At 10Hz and 0.1 proportional gain the system suffered the same offset as with 5Hz, as expected, however the oscillations during the process of striving to reach the target were of smaller amplitude and higher frequency. Convergence times of the 10Hz system were found to be only slightly lower than the 5Hz system with no real repeatable improvement other than the apparent stability around the target position when stationary. The 10Hz controller never required restraining or resetting physically between tests until after 0.3 proportional gain, where oscillations began tending to diverge and a step change of 200 created a chaotic response. The initial time step between the camera registering a change of target and the manipulator acting upon it varies from 30ms to 80ms, although none longer than this were observed despite being expected within the 5Hz system. At 5Hz the expected delay was up to one sample period, 200ms.

Standard Ziegler-Nichols (ZN) tuning techniques were also applied, however the results of this were unsatisfactory resulting in very rapid initial movement which triggered the robot controller lock-out. As the proportional value was adjusted to reduce this motion, it became apparent that the combination of integral and differential terms caused violent instability at any level - this technique was abandoned.

6.4 Two Timescale Testing

The two timescale system utilised the slow controller developed and tested in previous chapters, yet calculated the fast controller output at a rate of 1kHz. This controller does not assess overall position of the manipulator, but distance from the previous slow-loop measured error in order to affect only the oscillations and minimise interference between the two control loops. In order for the two loops to work independently, the slower loop must operate considerably slower than the fast loop in order for the slow loop to be able to take an average trend in direction. The fast loop must operate at a rate higher than the fastest oscillation frequency, at least

twice as fast according the Nyquist–Shannon sampling theorem, however even faster is preferable. The rate of this controller is effectively limited to that of the camera subsystem, 350Hz, as no further sampling is available. This is sufficient for the test apparatus in this research but may be insufficient for more rigid link structures or more complex structures with multiple flexible links as these may combine to cause end effector oscillations at higher frequency than the individual links. Both the slow and fast controller periods are adjustable at the commandline, at the same time as the fast and slow controller gains.

As can be seen in Figure 6.10, the slow controller should ideally be following the dashed average motion line. The fast controller should be acting to reduce the shaded areas above and below the dashed line, without interfering too greatly with the slow controller motion. Ideally, the fast controller would remove all oscillation from the motion and the manipulator would follow the dashed line without error. In reality the sampling rate of the slow controller means that, to some extent, the shape of the oscillations will be incorporated into the slow control loop. Intuitively it can be seen that, with higher execution frequency, the slow control loop will follow the oscillations instead of producing an average motion which would impact severely on the overall motion of the manipulator. Clearly, the faster the execution of the fast control loop, the higher the probability that the oscillations can be removed.

Testing was performed with four slow-controller frequencies - 5Hz, 10Hz, 36Hz and 100Hz - in order to determine the effect of the slow controller frequency interaction with the oscillations. The majority of the work with two timescale control was performed with the 3.5kg mass as the oscillatory frequency of this arrangement was within the range of controllability of the robot hardware. Remembering the hardware controller acts at 36Hz, the oscillation frequency in the order of 4-5Hz and the 10:1 ratio rule of thumb for controlling adequately, these frequencies represent samples around the important frequencies.

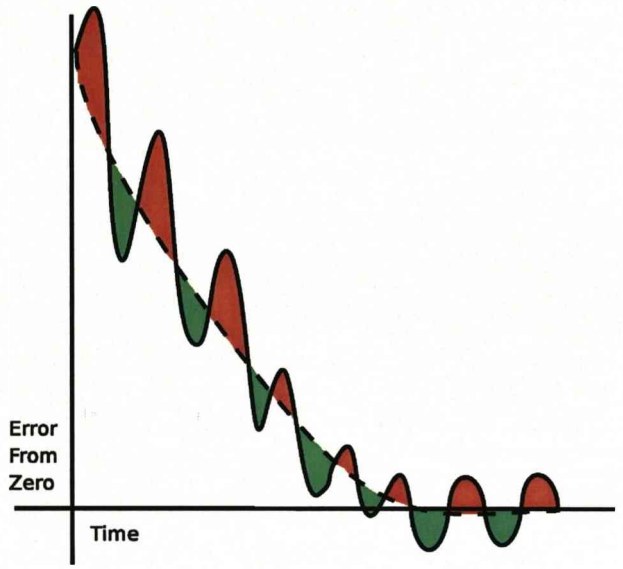

Figure 6.10: Diagram to show ideal behaviour of two timescale controller.

The process of testing the controllers to gather performance information was a systematic approach of initially fixing the slow controller proportional gain at a level that produces a the fastest response possible, with oscillation of the end effector, without introducing instability or such violent motion as to have the control terminated by the robot hardware controller. Differential gain was tested for each frequency but found to cause immediate instability when applied within the slow controller. Eventually, a figure of 0.3 proportional gain was found to work across the range of frequencies, to give the fastest response without robot lock-out. This fixed gain, on the slow controller, gave the chance to test the effects of the fast controller gains on an even ground later on.

6.4.1 5Hz Slow Controller

Starting at 5Hz it was immediately noticeable that, as with the single timescale testing, the two timescale method was susceptible to large inputs around the natural frequency of the flexible link. As with single timescale control, at at a proportional gain value of 0.3, the manipulator oscillated about the target position even at

rest, until negative differential gain was introduced. The oscillations stabilised at a differential gain of -0.75 and allowed a step input test to be performed from a steady initial state. Step tests at this point produced 4000ms primary convergence times. Progressing to higher negative gain reduced stability from this point. Introducing proportional gain to the fast controller accentuated the oscillations, particularly in the latter half of the step change, leading to longer convergence times in excess of 5000ms. Both the results and the physical motion of the manipulator lead to termination of the testing at 5Hz for concerns about damage to the equipment - they did not improve further with the adjustment of any gains. The best result achieved using the 5Hz controller, after extensive testing is shown in Figure 6.11. The rapid oscillations visible, as well as increasing convergence time, would rapidly destroy the drive mechanisms, this is an altogether unsatisfactory result.

6.4.2 10Hz Slow Controller

The same process of increasing the gains on the fast controller was employed on a 10Hz slow controller. Immediate improvements in system stability were evident at this frequency, due to the fact that the sampling time was not coincidental with the natural frequency of the mass-link combination. As can be seen in Figure 6.12, the oscillations could be well controlled and kept to a low amplitude during rapid end effector acceleration. This lead to a 1400ms primary convergence time, with secondary convergence by 1900ms. This is actually better performance than was achieved with the 0.3kg mass with a single timescale control architecture and higher proportional gains, Figure 6.6, with lower overall offset from the target position in addition. The final end effector oscillations, after secondary convergence, were reduced further than has been seen in any of the tests.

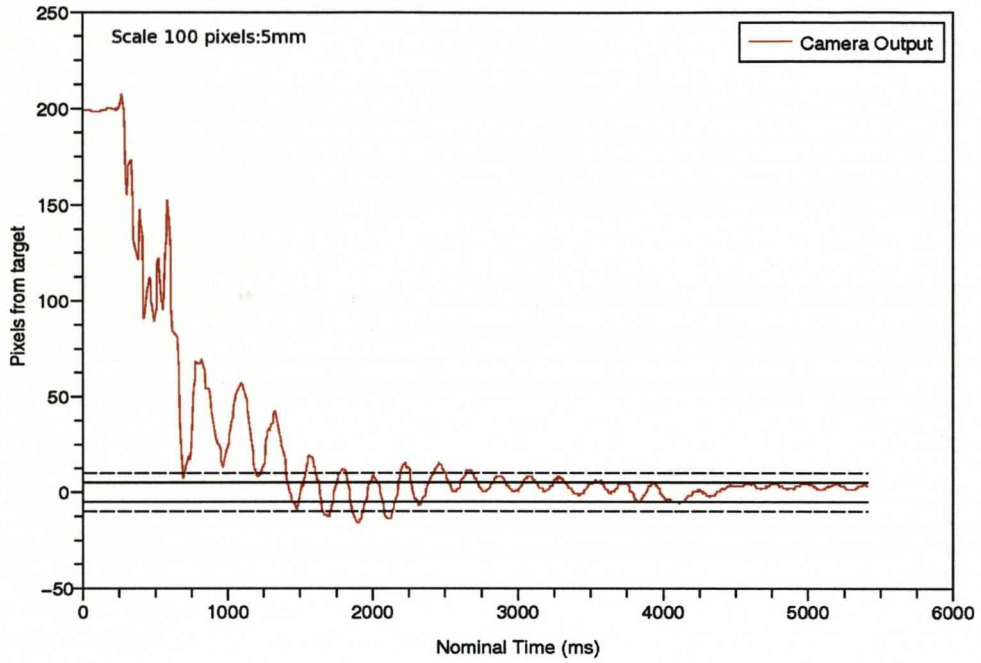

Figure 6.11: Optimum 5Hz response, PID Gains - Slow(0.3,0,0), Fast(0,0,-0.75).

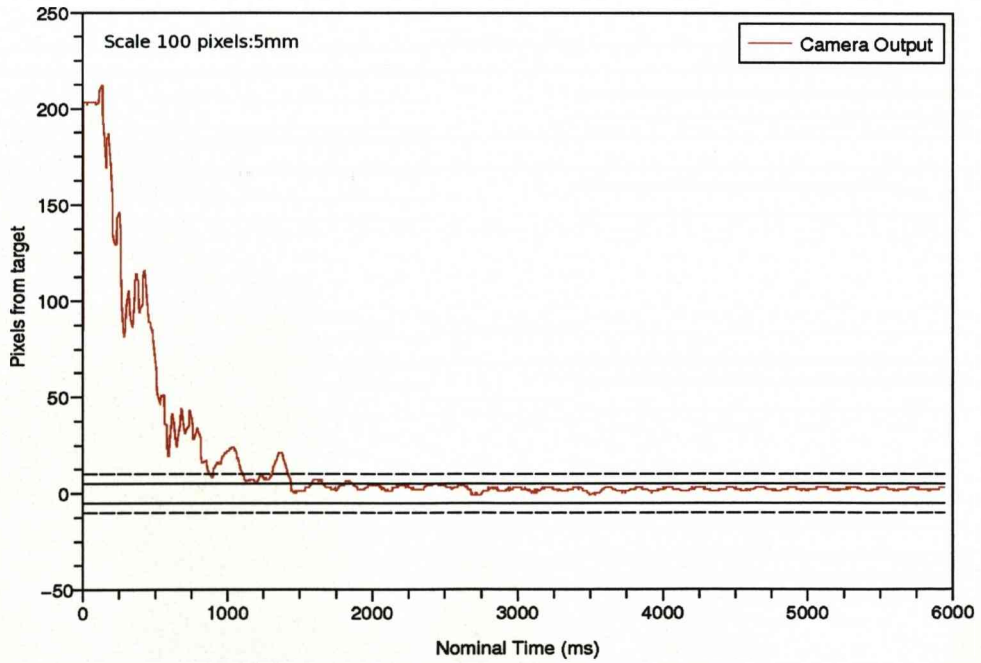

Figure 6.12: Optimum 10Hz response, PID Gains - Slow(0.3,0,0), Fast(0.2,0,-0.5).

6.4.3 36Hz and 100Hz Slow Controller

Increasing the slow control frequency above 10Hz resulted in lower performance overall. At 36Hz the slow controller clearly accentuates the link oscillations by attempting to follow these instead of the general trend towards zero as intended, as is visible in Figure 6.13.

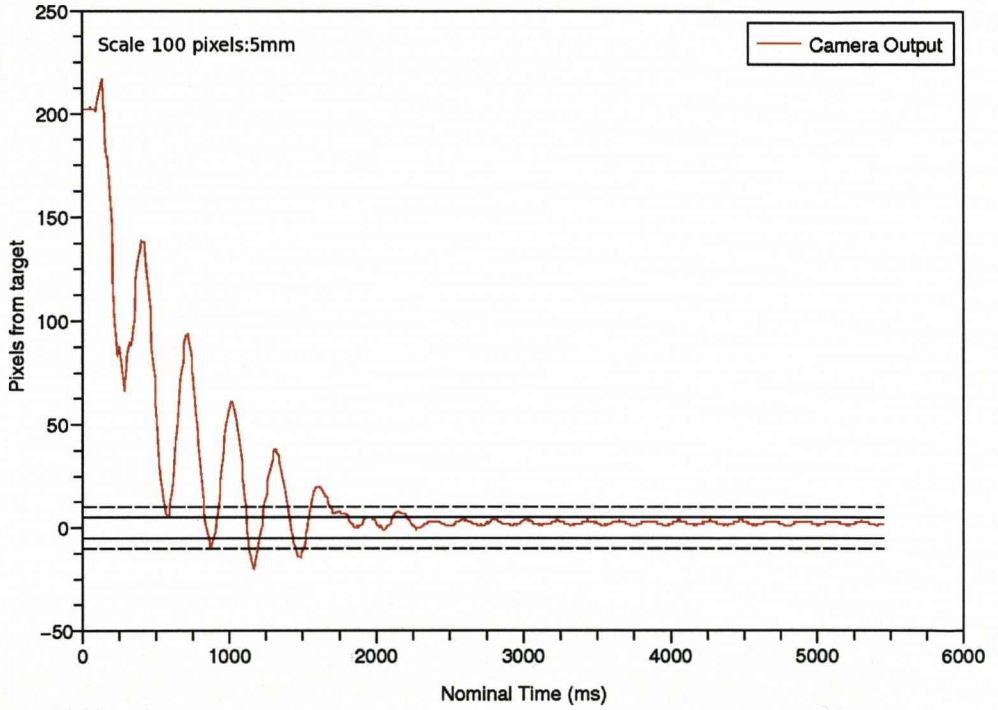

Figure 6.13: Optimum 36Hz response, PID Gains - Slow(0.3,0,0), Fast(0.2,0,-0.5).

At 100Hz there is so little difference from the 36Hz results, considering variability between tests, that they are indistinguishable from each other. Of interest is the fact that the 10Hz and 36Hz controller frequencies both appear to share the same optimum gain settings, whereas the 100Hz controller was far more susceptible to noise and only a tiny quantity of differential gain caused a very rough response. Neither of the these controller frequencies came close to the 10Hz controller in terms of rapid convergence and minimal post-secondary-convergence oscillation. These

results establish a trend in the response to control frequency inputs when using standard Proportional control in the slow control loop, but more importantly shows that it is possible to control the system's motion *and* oscillation effectively, using a single EIH camera feedback system, with the correct tuning.

6.4.4 Further Testing

With the stability added by the oscillation control of the fast controller loop, further tests upon the 10Hz control loop were carried out to determine if the response time could be improved further. It was found that the additional stability allowed application of derivative control on the slow controller, whilst simultaneously also allowing increased proportional gain to levels which were impossible without the fast controller. A sequence of tests were performed by iteratively increasing gains to instability, with the climax of control occurring at twice the slow proportional gain as was possible without the fast controller, and with a differential gain of -0.2. The results of this increase in slow gain can be seen in Figure 6.14, where the primary and secondary convergence times are 1210ms and 1260ms respectively. The higher proportional gain on the slow controller also delivers a smaller final offset, two pixels, representing a 0.1mm repeatable end effector placement despite initially unknown target placement.

6.5 Summary

Extensive testing was carried out on three control architectures, open loop, closed loop single timescale and closed loop two timescale. These were compared for settling time, elimination of oscillation both during and after the step change motion, and where possible, iterative gain adjustments. Iterative gain adjustments, after the application of two-timescale control, significantly improved the response time of the manipulator.

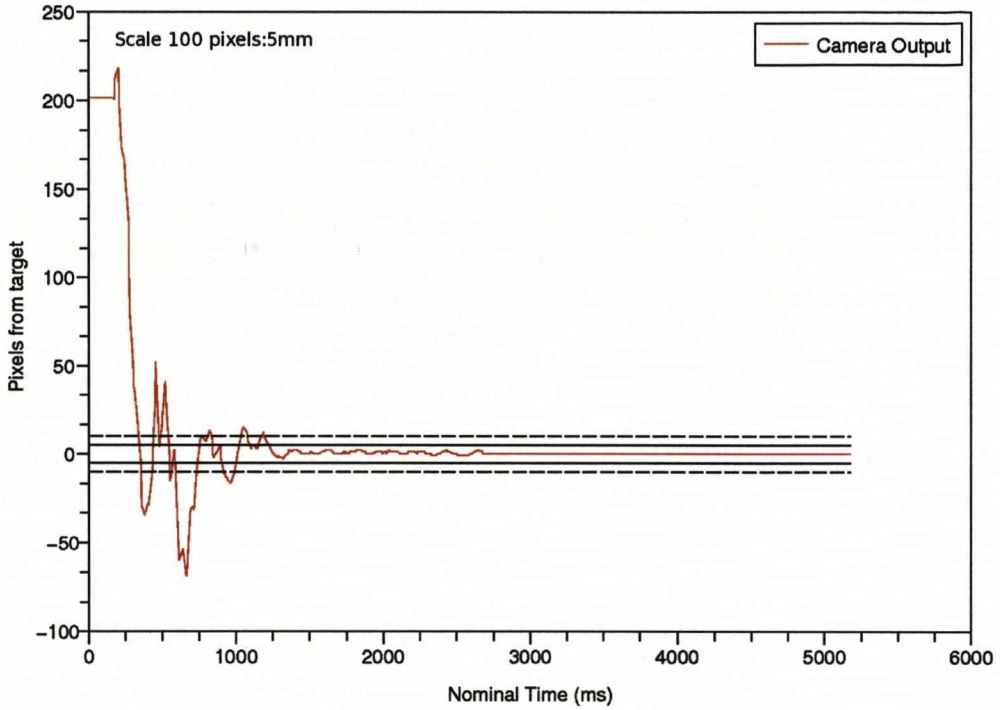

Figure 6.14: Optimum 10Hz response after iterative gain increase, PID Gains-Slow(0.6,0,-0.2), Fast(0.3,0,-1).

The introduction of differential gain in the slow controller, without first stabilising the oscillations with the fast controller, allowed rapid and high amplitude controller outputs to be generated. This causes rapid joint direction changes, forcing the SLAVE interface to cease communication with the Linux control software. This reduced the effectiveness of the single timescale controller, as the gains required to reduce end effector oscillation were too large to be applied to trajectory control without some differential gain to reduce initial acceleration.

A summary of the of the performance of the control architectures can be seen in Table 6.3. The table shows the primary (±0.5mm) and secondary (±0.25mm) convergence times. Undoubtedly the iterative gain adjustments on the two timescale system produced the best results due to the introduction of the negative derivative gain in the slow controller reducing the initial high acceleration. This, in turn,

reduces the final oscillation by reducing the strain energy stored in the link. The two timescale control reaches the high tolerance target in less than half of the time taken by the single timescale control. It has been shown that by careful analysis of the response of the manipulator to step inputs, a two timescale control method can be used to vastly improve the motion of the experimental system employed.

Table 6.3: Summary of control architecture performance with 3.5kg mass.

Control Method	$\pm 0.5mm$ (ms)	$\pm 0.25mm$ (ms)	Oscillation Control
Open–Loop	2750	2750	Very Poor
Single Timescale	1800	2500	Poor
Two Timescale	1400	1900	Good
Two Timescale Iterative	1210	1260	Best

Chapter 7

Discussion

7.1 Achievements

The main objective of this work was to assess the feasibility of using a visual servoing system to control a flexible manipulator with only the feedback provided by a single high speed camera and the basic joint encoders. One of the aims of this research was to provide a route to allowing the cost of manufacturing manipulators to be reduced by permitting the use of low rigidity links and lower quality drive systems. The target application of the manipulator was to be robotic welding, an area increasing by an average of 12% per annum over the last decade and where robots are increasingly favoured due to a growing lack of skilled workers.

A purposefully degraded manipulator was created, along with image processing software and the two brought together by creating a central control computer software environment. The system was to capable of assessing the abilities of a high speed visual servoing system with minimal sensor use, in controlling a hard-to-model and varying manipulator system without detailed models, leading to future investigation into improving the control strategies and sensors. The following subsections summarise the achievements in each section of the task, before discussion of the

results and how system performance have been improved, then finally conclusions and future work will be provided in the following chapter, Chapter 8. The original objective of the work was fully achieved and has opened the way to the design and control of a compliant robotic system using a single EIH high speed camera system.

Vision

In order to achieve the project objective a Windows^{TM} based, high speed computer vision system was created. It uses an inexpensive CMOS-based machine vision camera mounted at the end effector of the robot manipulator to give direct measurement of end-effector error - an eye-in-hand system. Image processing software was developed for the implementation of pattern recognition algorithms, to gather the information required to test the control system performance. This uses rapid thresholding and edge detection algorithms, combined with the hardware's windowing capability to provide high framerates. The software assumed a seam following form and was able to operate at 500Hz in standalone mode. Due to the Windows-only driver requirements, the system had to transmit its feedback data over a standard network link to the central control computer, this reduced its performance to 350Hz. This was sufficient to use as an experimental setup where the camera system provided image based feature extraction in the form of manipulator-tip error from target.

In order to transmit the data over the network link, a server-client software system was developed and connected over dedicated ethernet hardware. The images were processed on the Windows^{TM} machine, relevant information extracted and passed over the dedicated link to the server software on the control computer, which both timestamped and logged the information in real-time, while passing it to the control software simultaneously.

Robot

A PUMA 560 robot arm was retrofitted with a performance degrading flexible link to reflect a manipulator designed with strength requirements, rather than rigidity requirements. The robot backlash could also be de-adjusted to simulate poor drive component quality if required and so serves as a suitable testbed.

The flexible link was designed and constructed; as expected it gave large oscillations at a rate that was controllable with the PUMA hardware, despite handling a payload greater than that specified for the robot. The mechanical design assumptions were tested and proved. Using the control software, the robot was limited to three degrees of freedom in order to perform linear motion tests with an array of control architectures. The control system was limited to three degrees of freedom in order to prevent drift or interference from the other joints during the specific tests applied. These tests had no feedback to control drift in the other axes of motion and rotation and therefore it was considered best to immobilise joints one, four and six until the visual subsystem was developed further to provide feedback for these variables. It was, however, capable of fully controlling the other joints should future testing and control architectures demand it.

Control

At the heart of the task lay the requirement to develop a system capable of combining the camera feedback and joint position data, then using it to accurately and repeatably place the manipulator, despite the un-modelled end effector deflections. In order to do this, available operating systems were first investigated for performance and suitability to the task at hand. RTAI, a real-time patch to the conventional Linux kernel was selected after consulting both academic literature and users of the various options. This was then implemented on a standard desktop, AMD Athlon based computer with no special hardware requirements.

The robot interface, SLAVE required a driver to be created in order to allow the robot to communicate with the computer. This was created in a manner that could be removed from the system later and replaced by drivers for specific analogue interface hardware for standalone robot control without the SLAVE interface. The robot control hardware communicates in raw joint angles, with the hardware controller applying the required control loops and gains to attain and maintain the commanded joint position in the fastest possible time.

A central, hard real-time, control thread was created as a vehicle by which various control techniques were tested. This used several methods of inter-process communication and deadlock avoidance in order to operate in hard real-time whilst communicating with both the user interface, the robot hardware and the camera data server. The control thread allowed multiple control loops to run at different frequencies, and the commandline interface allowed controller gain and frequency setting while online, with a view to being able to use adaptive control to automatically set these parameters.

7.2 Open Loop Control

Open loop control systems are how many industrial robotic systems function, relying on high quality models of the manipulator and process to position the end effector in the correct location. Naturally, if any part of this process or manipulator is different to that assumed in the model, the open loop control system will fail to position the arm correctly or within a usable time.

The open loop tests show the vast change that occurs when some part of the process or manipulator is adjusted. A simple payload change from 0.3kg to 3.5kg was applied and the effect on the manipulator's performance was assessed. With the lower load the manipulator did a reasonable job of positioning the end effector in a very short time, though with some oscillation. When that load was increased,

the manipulator's operating speed had to be reduced by a factor of 200 in order to prevent the flexible link from causing damaging oscillations. Even then, the settled response time was significantly longer and had more residual small-scale oscillations. Although methods exist for adapting model based control, online, by learning the manipulator dynamic parameters, they still rely on the assumption that the robot links are rigid and the drive gears are high quality.

The open loop testing did show the repeatability in the basic manipulator, with the same position being reached on each repetition. This matches the robot's quoted repeatability of $\pm 0.1mm$. Testing also showed a non-linear relationship between VAL speed setting and actual linear speed of the manipulator end effector.

7.3 Closed Loop Control

Closed loop control of the manipulator was tested in two main guises, single and two timescale. Testing of the manipulator began with single timescale control, where the kinematics of the robot calculated the current assumed position of the robot - that is the position assuming all the links and drives are rigid and backlash free. The PID control loop then compared this assumed position to the image data, showing error from target, and amalgamated the data to produce an adjustment to the commanded position. This updated position was then transmitted to the robot by calculation of the joint angles using the inverse kinematics, again assumed rigid.

7.3.1 Single Timescale Control

It was hoped that the high rate of control updates would allow accurate measurement of the end effector displacement and allow the target to be reached with minimal oscillation, in a short time. With high proportional gains, the system responded very rapidly, too rapidly. The step change in joint position caused the robot's hardware

controller to lock out the joints to prevent damage due to drive current demands exceeding what the hardware is capable of delivering.

Ziegler–Nichols tuning methods were attempted on the single timescale PID routine but were not successful as joint hardware torque limitations were exceeded before steady *controlled* oscillation could be achieved with the required 3.5kg payload. Instead, a structured, iterative testing process was employed to determine the optimum gain settings for P, I and D terms. Even small integral gains caused large integral wind-up, initially, when the integral term was measured from all previous samples. Shorter integral calculation windows with lower gains were employed with some improvement but then were not sufficient to remove the proportional offset.

The proportional offset in the Cartesian image-space is due to the addition of offsets in each joint position. Each joint strives to reach its desired target but is unable to do so perfectly, in most cases, due to joint friction and per-joint controller gains (set in the robot hardware) being insufficient for the payload applied. If the robot system was not controlled through the SLAVE interface and associated hardware, these gains could be tuned more precisely - currently they are set to operate with the VAL dynamic assumptions and normally alter with the selected VAL speed setting.

In an attempt to reduce the initial acceleration of the manipulator, to one that does not cause joint lock-out, some negative differential gain was introduced. The purpose was to see the large initial change in target error and tame the proportional control for its first few samples. Unfortunately the initial motion of the manipulator causes the camera to swing and point in a direction opposite to the overall direction of travel. This initial peak, as visible in all of the figures in Chapter 6, and subsequent oscillations caused very large D term output. This caused great instability, even slight disturbances introduced purposefully, caused erratic behaviour. Further investigations were required.

7.3.2 Two Timescale Control

It was observed that the manipulator manifests two types of motion while target tracking in open-loop mode. There is a so called rigid response, which takes the form of the general motion of the manipulator, and then superimposed upon the rigid response is the flexibility of the compliant link, or links. After investigation into the single timescale controller, it was acknowledged that the high frequency oscillatory response of the flexible link was interfering with the overall motion of the manipulator and increasing the response time. The effect of the oscillations on the single timescale controller meant that the advantageous use of differential gain could not be implemented to slow the initial movement and reduce the oscillations.

The control calculations were separated into two control tasks, working at differing execution and sampling rates. This way the overall motion of the controller could be controlled by one set of gains using a "slow controller", while the oscillatory motion could be removed by gains suitable for that task, executed by a "fast controller". The two control loop outputs would were then superimposed and applied to the manipulator in a feedback loop. Both control loops took parallel PID form but the slow controller used raw error feedback for its calculations, while the fast controller used the distance of the latest camera position from the last slow controller sample. This meant that the fast controller was always striving to reach the last slow controller sample, which could be considered to be the mean position. Initially the slow controller had been implemented with a low-pass filter on the input, created by averaging many of the last input samples to get the general trend of motion. Unfortunately this proved to be less successful than normal PID control that was finally implemented. This is because the averaging produced large settling times and long delays in initial movement, due to the oscillations demanding many samples to be averaged in order to follow the general trend of motion.

The final method of implementation, within RTAI, allowed the online adjustment of the controller gains and execution rates for both the slow and fast controllers by simple commandline interface. A process of iterative tuning was employed to slowly increase proportional gain on the slow controller to the point where the step changes were too great for the robots hardware controller to drive. The form of the response was analysed and the gain with the fastest settling times was selected for further experimentation. Working from this point, the fast controller gains were tested, first with increasing proportional control, but then also with differential control. As described by Figure 5.12 and its accompanying text in Chapter 5.3.4, negative differential gain tends to temporarily reduce the action of the manipulator. This effect tends to slow the manipulator movement if the camera and payload are lagging behind, and speed up the manipulator if they are leading. This is better explained by Figure 7.1, with time samples t_1 and t_2. These time samples represent the points at which the manipulator end effector oscillatory speed is at maximum positive and maximum negative speed, in relation to the slow controller's last recorded position, respectively. As can be seen from the diagram, the negative differential gain produces a maximum negative fast controller output at t_1 and maximum positive at t_2. This acts to slow the oscillation and bring it to towards the average motion of the end effector.

Proportional gain was also increased independently of the differential gain on another series of tests and was found to simply create a instability, with the two control loops fighting each other and making the arm oscillate further. This occurred on all tests with the two-timescale arrangement. The negative differential gain produced the most stable and ideal response in each case, up to a fairly constant gain of -0.1, at which point the response began to degrade again. This is the point where the differential gain begins to overrule the slow controller, and the controllers interact in an undesirable fashion.

The first tests were performed at a relatively high, 100Hz, though this was determined to be not ideal as this frequency of sampling allows the slow controller to follow the oscillatory response, not the rigid response. For this reason a range of tests were carried out, starting at the natural frequency of the oscillating beam, passing through $2\omega_n$, and the hardware control frequencies. These were to determine the effect that each of the sampling rates had upon the rigid response of the manipulator. At 5Hz the controller presented problems that are associated with low frequency control of rigid structures, the time between samples was insufficient to produce a smooth motion of the robot, leading to a stuttering response as each update modified the manipulator position by a large distance. This was eased somewhat by the differential nature of the fast controller, however its performance was lower than that of single timescale control.

The 10Hz slow controller provided a smooth and fast response across the same range of gains tested, and allowed higher proportional gain to be applied than to the 5Hz controller. This was because the time between samples and updates was halved and therefore the robot hardware controller had to command a smaller difference

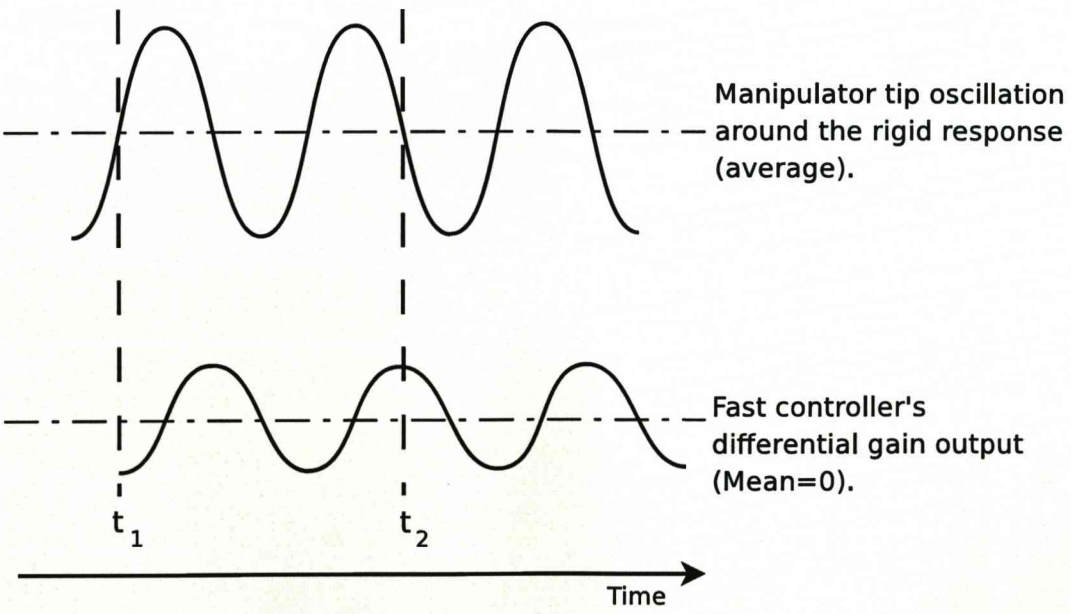

Figure 7.1: Fast controller output, in relation to end effector oscillation about mean.

between joint angles. Also, with the 5Hz controller operating very close to the natural frequency of the beam, the controller tended to sample at alternating sides of the rigid response, this caused a bang-bang type of controller which does not work with fast reacting, essentially un-damped oscillatory systems. It was found that a small quantity of proportional gain in the fast controller helped to stabilise oscillations after secondary convergence. This is due to the fact that the negative differential gain produces controller outputs proportional to the linear speed of the oscillation. At low oscillation amplitudes, that linear speed is reduced and the effect of the controller is also reduced - at this point a small proportional gain corrects the small errors without dramatically affecting the overall operation of the fast controller.

The 36Hz and 100Hz controllers both followed the oscillations of the end effector too accurately and simply added to the instability of the system when the slow controller was at a gain setting similar to those in the10Hz tests. The introduction of negative differential gain, as per the 5Hz and 10Hz tests, did not produce significantly improved response, although some improvement was apparent.

Due to time constraints, tests on slow controller frequencies from 5Hz to 100Hz in 1Hz steps could not be achieved, however the 10Hz was selected as the best results due to its stability and minimal oscillations when tuned with fast negative differential gain. In an iterative process again, the slow proportional gain was increased further while the oscillatory action was controlled by the fast controller. This worked again, to a level much higher than before, to the point where the step response by the slow controller was so great that the robot hardware controller locked out again. At this point, with the oscillations still under control by the fast controller, negative differential gain was introduced into the slow controller to reduce the initial step rate while maintaining a higher proportional gain for faster overall response. This allowed much larger increases in slow proportional gain to be applied which drastically reduced the primary and secondary convergence times. At these higher

slow controller gains it was found beneficial to increase the fast differential gain slightly also, to deal with the slightly larger end effector velocities. The secondary convergence time was reduced by a further 50% over the original two-timescale, 10Hz, controller.

7.4 Summary

Examining the results as a whole, it is apparent that the two timescale control provides good response to a step input, despite having an inaccurate, rigid, kinematic model of the compliant robot. The control system's sampling speed is sufficient to calculate its output to help minimise oscillatory action while still making the bulk movement to the target in a short period of time. With the final iterative solution, the system settled to within $\pm 0.25mm$ in just 1260ms with very little oscillation after this point. The system settled to the same position upon each test at the chosen gains, showing very high repeatability despite kinematic anomalies. The limiting factor in the two timescale PID control is the resonant frequency of the beam. The slow controller must be executed at at least twice the frequency of resonance of the beam and payload, but not significantly higher because it is then liable to following the oscillatory motion of the end effector.

Although not providing the *optimum* control solution, the two timescale control proved that high speed camera feedback can be used, without dynamic modeling or additional sensor systems, to provide effective control over compliant robotic structures, proving the feasibility of reducing the cost of manipulator hardware.

Chapter 8

Conclusions and Future Work

Accepting the details of the construction and testing of the system developed in the research and taking an overview, this chapter provides a view of what was achieved during the whole process, as well as highlighting areas needing further research. During the assessment of the possibility of compliant visual control, the following objectives were achieved:

- A fully working robot control system was developed for the six DOF manipulator, scalable from one to six (or more) joints, and in a modular format allowing hardware changes. This provides the opportunity to transfer the control system to different robot hardware with minimal code modification and computer hardware changes. The system was developed using an open source operating system and utilised a single, remote, camera.
- It was also shown that despite a complex, non-linear, degraded performance manipulator, a single high speed eye in hand vision system can be used, without dynamic modeling of the manipulator, to provide reasonable control over the system.

- A simple, fast, accurate image processing and logging system was created and operated at 350Hz to give high speed measurement of end effector placement. The feedback from the camera was used to provide both positional control and oscillatory damping, in a two timescale control system utilising basic PD control techniques and no sensors other than the joint encoders in the manipulator.
- This image-based visual servoing system is capable, with little modification, of operating as a full standalone robot controller without the SLAVE digital/analogue interface system.
- A stable basis for further experimentation in the visual servoing field has been created.

8.1 Hardware Design Limitations

Despite creating a working compliant robot visual controller and testing it with a realistic situation, there were compromises made during the design and testing that could not be overcome with the restrictions on time and cost. The primary cause for concern was the remote camera system, communicating with the controller over a standard network link. The camera system frame rate and rate jitter is primarily determined by the computational power of the host Windows™ XP system, which is a non-realtime system. Being a non-realtime system it is designed primarily for average throughput of all processes running at the time.

Windows™ XP system processes are often higher priority than so called user processes, leading to preemption of the user processes and delayed or slowed computation. The camera software reports average framerate with an update every second. Although the average framerate varied very little, actual frame rates could be seen to be significantly slower at times of high processor load, dropping to around 250Hz. Also, within the 350Hz average frame rate, the camera frames could be

transmitted in bursts with unpredictable pauses. This poses severe problems for any control system and, although these pauses are known quantities recorded at the controller end, they were only used as logs and not integrated into the control strategy. The delays and variations in visual processing times were accentuated by having to transmit the data over a network link which, although the fact that it was a dedicated link and operating away from its maximum capacity would suggest that this was not likely to add significant error.

If the camera system has been incorporated into the Linux/RTAI control system directly these problems would have been minimised and the image data processed in a low latency, low jitter system. No camera system available at the time of the project had Linux interface capabilities and PixeLINK would not provide their proprietary driver information to allow the creation of such drivers from scratch.

The PUMA manipulator has severe joint torque limitations, as well as a low SLAVE interface speed in comparison with modern manipulator systems. Neither of these is conducive to controlling a high speed oscillatory system, however these were still able to perform well enough to control a low-speed oscillation and positioning system, which is therefore scaleable. Using faster analogue output driver boards linked directly to the control computer, a manipulator with higher torque joint motors (possibly direct drive) and suitable control strategies, it is believed that this work shows that the high speed eye in hand system is capable of full control at higher frequencies and with more than one flexible links.

8.2 Software Design Limitations

The system produced was designed purely as proof of concept, as an investigative tool, in this task it performed correctly. The control strategies were sub-optimal and dependant on tuning that did not follow any known tuning method, however the process of tuning these controllers provided a general strategy for refining the control

based on the previous results. The two-timescale control system would have been more successful if the slow controller had not relied on instantaneous measurements of displacement which were susceptible to following the oscillatory motion of the flexible link. A form of trajectory planning would be more effective in the slow control loop, with the high speed controller modifying the pre-planned motion.

The three distinct sampling rates provided non-coincidental data updates, this creates unpredictable delays in the control structure leading to further unpredictability in the control system, this presented itself as occasional erroneous results.

8.3 Future Work

In order to progress further, there are some areas that should be recommended for further investigation. It has been highlighted that the SLAVE interface operates at a low refresh rate of 36Hz, this interface is too slow for effective control of systems with resonance above that investigated in this research. More could be achieved if the PUMA and its hardware interface were to be replaced by custom hardware. Likewise, the control computer side of the interface could be modified to provide direct control of the manipulator motors. This way the full potential of the control computer could be investigated without constraints applied by the interface system.

The second area that should be investgated further is the use of adaptive and intelligent, non-model based control algorithms. The iterative method of tuning the controllers used in this research is clearly not applicable to industrial use. Although the two timescale nature of the manipulator was effectively controlled using basic PID controllers, much larger improvements could be realised with the use of self-teaching architectures. These would be applicable within the industrial environment and as they have been proven on rigid robots, they are a suitable next step in compliant robotic research.

Intelligent Control Methods

Although PID control is useful and can be robust in control of simple systems it is not ideal for complex nonlinear systems due the breakdown of the linear approximations and varying parametric data throughout a process. Model based control also suffers from degradation of performance when items not considered in the model are introduced. In order to overcome these problems adaptive and intelligent control methods are employed, these are methods that either do not rely on a model, or sense parameters of manipulator or process to use in a model.

Again, consider the ability of biological control systems such as the human brain to control non-rigid manipulators with varying payloads and unpredictable environments, in remarkably accurate and rapid manner. This control is considered to be based on a set of inexact but descriptive control functions, several of which could be true at any one instant - "manipulator is moving quickly and oscillating so reduce oscillation". The amount to which each of these functions is considered in the control system is determined by a weighting function - a measure of its importance in the current situation, based on loosely defined rules. The precise output of this control system depends on the result and weighting of the active functions.

This method of control was developed by L. Zadeh in 1964 [78], and subsequently named Fuzzy Logic (FL), due to the imprecise and descriptive nature of the control functions. FL has remarkable abilities to control complex systems without the need to develop models or set strict parameters on the control functions, making FL ideal for control of difficult to model hardware. Fuzzy systems, being based on simple logical tests and weighting functions, are easily implemented on a low power computer system, to such an extent that they are used in diverse applications from pattern recognition for optical character recognition to elevator control.

Further gains in performance can be achieved by many methods of combining fuzzy control with a learning procedure to further refine the weighting functions used

in the system, based on previous action [79]. When considered within the context of a process that is repeated and relatively similar, neural networks can be trained to provide optimal outputs for a given situation. The combination of the two methods as a neuro-fuzzy system offers the possibility of a highly adaptable, highly accurate control system that is easily implemented on a rapid, real-time control computer - some examples are considered below.

Intelligent methods can be applied to the field of manipulator control, [80] uses neural networks to identify and control an unknown motor system with success. Camera calibration using simple learning techniques has been used to increase the effectiveness of visually servoed robots, particularly in image based servoing [81]. A two-part PD and adaptive dynamic system is used by [82] to control a two DoF manipulator with the adaptive control compensation algorithm estimating and applying the dynamic effects of payload and manipulator within the first 500ms of movement. The process of using the feedback to learn the parameters of the robot, online, would benefit from visual information as a direct measure of performance instead of estimated performance from joint angle information. Xiaoyu [83] demonstrates a new visual servoing scheme whereby a fuzzy controller is used to move the manipulator into the approximate position of the task, followed by a neural network controller that exactly positions the end effector. Stanley et al. presents a hybrid visual-kinematic solution [84] which combines visual servoing and direct computed kinematics using a neural network system, a simple PD controller was used to obtain the final end effector positioning. Ultimately, Wai [85] introduces design and analysis of a four layer neural fuzzy network, parameter estimation controller for n-link manipulators. This is applied to a two link rigid manipulator and reduces joint tracking errors from around 2.5 degrees to less than half a degree within one second of manipulator motion, while tracking a sinusoidal target. Although this kind of joint position correction is not useful in a flexible system, because joint positions do not necessarily relate to end effector position, the strategy of estimation and

direct application of corrections could be used in future work to vastly improve the manipulator control without needing to model the system. This would allow a manipulator to be used with changing environments and payloads without requiring calibration or re-tuning manually.

This work considered only a single flexible link at the tip of the rigid manipulator, the addition of other flexible links introduces more complexity in the kinematic representation of the robot. In a real situation there could be five or more flexible links, each of which is providing some uncertainty in the kinematic calculations. In order to counteract this, the process of determining which of the joints should be moved and by how much becomes significantly harder to determine. A process of iterative joint position modification, with a joint priority system based on the current robot pose and that joint's direction of action, is a solution worthy of further work. This work is out of the remit of this investigation but is a logical next step in the research.

References

[1] R. Middleton, J. Ward, J. Freudenberg, and A. Woodyatt, "Performance limitations in the feedback control of a class of resonant systems," in *Decision and Control, 1999 Proceedings of the 38th IEEE Conference on*, vol. 2, pp. 1845–1850, 1999.

[2] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 77, pp. 215–221, 1955.

[3] X. Jiang, A. Konno, and M. Uchiyama, "Visual servoing experiment using a 3D flexible-link manipulator," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 1224–1229, 2006.

[4] X.-J. Shen and J.-M. Pan, "A simple adaptive control for visual servoing," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 2, pp. 976–979 Vol.2, 2003.

[5] S. Han, W. See, J. Lee, and H. Hashimoto, "Image-based visual servoing control of a SCARA type dual-arm robot," in *Industrial Electronics, 2000. ISIE 2000. Proceedings of the 2000 IEEE International Symposium on*, vol. 2, pp. 517–522 vol.2, 2000.

[6] R. Kelly, E. Bugarin, I. Cervantes, and J. Alvarez-Ramirez, "Monocular direct visual servoing for regulation of manipulators moving in the 3D cartesian

space," in *Decision and Control, 2006 45th IEEE Conference on*, pp. 1782–1787, 2006.

[7] E. Malis, F. Chaumette, and S. Boudet, "2.5D visual servoing," *Robotics and Automation, IEEE Transactions on*, vol. 15, pp. 238–250, 1999.

[8] P. Corke, "An experimental facility for robotic visual servoing," in *TENCON '92. "Technology Enabling Tomorrow : Computers, Communications and Automation towards the 21st Century.' 1992 IEEE Region 10 International Conference.*, pp. 252–256 vol.1, 1992.

[9] A. Hafez and C. Jawahar, "Probabilistic integration of 2D and 3D cues for visual servoing," in *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, pp. 1–6, 2006.

[10] A. Ranftl, L. Cuvillon, J. Gangloff, and J. Sloten, "High speed visual servoing with ultrasonic motors," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4472–4477, 2007.

[11] V. Vyawahare and N. Afzulpurkar, "Uncalibrated eye in hand visual servoing for fixtureless assembly automation," in *Industrial Technology, 2006. ICIT 2006. IEEE International Conference on*, pp. 2244–2249, 2006.

[12] D.-W. Lee, S.-H. Kim, and S.-C. Won, "Visual servoing method using camera self-calibration," in *SICE-ICASE, 2006. International Joint Conference*, pp. 5006–5010, 2006.

[13] P. J. S. Goncalves, L. F. Mendonca, J. M. C. Sousa, and J. R. C. Pinto, "Uncalibrated eye-to-hand visual servoing using inverse fuzzy models," *Fuzzy Systems, IEEE Transactions on*, vol. 16, pp. 341–353, 2008.

[14] G. Chesi and K. Hashimoto, "A self-calibrating technique for visual servoing," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 3, pp. 2878–2883 vol.3, 2002.

[15] E. Snow and D. Yoerger, "Devising a misalignment tolerant subsea grasping system," in *OCEANS '97. MTS/IEEE Conference Proceedings*, vol. 2, pp. 1222–1229, Dept. of Mech. Eng., MIT, Cambridge, MA, USA, October 1997.

[16] W. Jongkind, "Dextrous gripping in a hazardous environment, guidelines, fault tolerance and control," in *Systems, Man and Cybernetics*, vol. 1, pp. 509–514, Delft University of Technology, Netherlands, October 1993.

[17] J. Mills, "Manipulating rigid payloads with multiple robots using compliant grippers," *Mechatronics, IEEE/ASME Transactions on*, vol. 7, no. 1, pp. 23–34, 2002.

[18] J. Hill and W. T. Park, "Real time control of a robot with a mobile camera," in *9th International Symposium on Industrial Robots*, pp. 223–246, March 1979.

[19] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *Robotics and Automation, IEEE Transactions on*, vol. 12, pp. 651–670, October 1996.

[20] N. Hollinghurst and R. Cipolla, "Uncalibrated stereo hand-eye coordination," *IVC*, vol. 12, pp. 187–192, April 1994.

[21] S. Wijesoma, D. Wolfe, and R.J.Richards, "Eye-to-hand coordination for vision-guided robot control applications," *International Journal of Robotics Research*, vol. 12, pp. 65–78, February 1993.

[22] B. Yoshimi and P. Allen, "Active, uncalibrated visual servoing," in *1994 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 156–161, 1994.

[23] A. Muis and K. Ohnishi, "Eye-to-hand approach on eye-in-hand configuration within real-time visual servoing," in *The 8th International Workshop on Advanced Motion Control*, pp. 647–652, March 2004.

[24] R. K. Lenz and R. Y. Tsai, "Calibrating a cartesian robot with eye-in-hand configuration independent of eye-to-hand relationship," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 916–928, September 1989.

[25] G. Flandin, F. Chaumette, and E. Merchand, "Eye-in-hand / eye-to-hand cooperation for visual servoing," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2741–2746, 2000.

[26] P. I. Corke, "Visual control of robot manipulators - a review," tech. rep., CSIRO Division of Manufacturing Technology, 1992.

[27] P. Corke, *High-Performance Visual Closed-Loop Robot Control.* PhD thesis, University of Melbourne, Dept. Mechanical and Manufacturing Engineering, July 1994.

[28] L. Bascetta and P. Rocco, "Task space visual servoing of eye-in-hand flexible manipulators," in *Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, vol. 2, pp. 1442–1448, July 2003.

[29] L. Bascetta, *Visual Servoing of Flexible Manipulators.* PhD thesis, Politechnico di Milano, 2004.

[30] L. Bascetta and P. Rocco, "Two-time scale visual servoing of eye-in-hand flexible manipulators," *IEEE Transactions on Robotics*, vol. 22, pp. 818–830, August 2006.

[31] A. B, O. Khatib, and J. Burdick, "The explicit dynamic model and inertial parameters of the PUMA 560 arm," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 510–518, 1986.

[32] S. Eppinger and W. Seering, "Introduction to dynamic models for robotic force control," *IEEE Control Systems Magazine*, vol. 7, no. 2, pp. 48–52, 1987.

[33] L. Weiss, *Dynamic Visual Servo Control of Robots: an Adaptive Image-based Approach.* PhD thesis, Carnegie-Mellon University, 1984.

[34] Z. Qiu, "Acceleration sensor based vibration control for flexible robot by using PPF algorithm," in *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*, pp. 1335–1339, 2007.

[35] J. Pomares, F. Chaumette, and F. Torres, "Adaptive visual servoing by simultaneous camera calibration," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2811–2816, 2007.

[36] T. Singaraju, A. Turan, M. Gokasan, and S. Bogosyan, "Hardware-in-the-loop simulation of PUMA 560 via internet," in *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, pp. 5426–5432, 2006.

[37] M. Khelfi and A. Abdessameud, "Robust h-infinity trajectory tracking controller for a 6 D.O.F PUMA 560 robot manipulator," in *Electric Machines & Drives Conference, 2007. IEMDC '07. IEEE International*, vol. 1, pp. 88–94, 2007.

[38] W. Dixon, D. Moses, I. Walker, and D. Dawson, "A simulink-based robotic toolkit for simulation and control of the PUMA 560 robot manipulator," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4, pp. 2202–2207 vol.4, 2001.

[39] P. Corke and B. Armstrong-Helouvry, "A search for consensus among model parameters reported for the PUMA 560 robot," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 1608–1613 vol.2, 1994.

[40] V. Becerra, C. Cage, W. Harwin, and P. Sharkey, "Hardware retrofit and computed torque control of a PUMA 560 robot updating an industrial manipulator," *Control Systems Magazine, IEEE*, vol. 24, pp. 78–82, 2004.

[41] P. Mantegazza, E. Bianchi, and L. Dozio, *DIAPM RTAI Programming Guide 1.0.* Dipartimento di Ingegneria Aerospaziale - Politecnico di Milano, 2000.

[42] A. Harnesk and D. Tenser, "Real-time performance of windows XP embedded," *http://www.idt.mdh.se/utbildning/exjobb/files/TR0470.pdf*, 2006.

[43] M. Timmerman and M. J. C., *Windows NT as real-time OS?*, vol. 5 of *Real-time systems conference*, pp. 573–586. Teknea, 1997.

[44] T. Bird, "Comparing two approaches to real-time linux," *http://www.linuxdevices.com/articles/AT7005360270.html*, 2000.

[45] D. Aarno, "Autonomous path planning and real-time control – a solution to the narrow passage problem for path planners and an evaluation of real-time linux derivatives for use in robotic control," Master's thesis, Royal Institute of Technology, Sweden, 2004.

[46] P. Masarati, R. Bucher, H. Mayer, L. Dozio, D. Schleef, D. Gasperini, P. Soetens, P. Mantegazza, M. Neuhauser, and G. Racciu, "RTAI - the RealTime Application Interface for Linux from DIAPM," *http://www.rtai.org.*

[47] E. Paljug, T. Ohm, and S. Hayati, "The JPL serpentine robot: a 12 DOF system for inspection," in *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3143–3148, May 1995.

[48] G. Chirikjian and J. Burdick, "A hyper-redundant manipulator," *IEEE Robotics and Automation Magazine*, vol. 1, pp. 22–29, December 1994.

[49] J. Ostrowski and J. Burdick, "Gait kinematics for a serpentine robot," in *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1294–1299, April 1996.

[50] P. Benham, R. Crawford, and C. Armstrong, *Mechanics of Engineering Materials.* Prentice Hall, 2 ed., 1996.

[51] P. McKerrow, *Introduction to Robotics.* Addison-Wesley Publishing Company, 1998.

[52] Unimation Inc., Shelter Rock Lane, Danbury, CT 06810, *PUMA Robot Equipment and Programming Manual (Model 550/560)*, May 1982.

[53] P. J. W. Noble, "Self-scanned silicon image detector arrays," *IEEE Transactions on Electron Devices*, vol. 15, no. 4, pp. 202–209, 1968.

[54] S. Chamberlain, "Photosensitivity and scanning of silicon image detector arrays," *IEEE Journal of Solid-State Circuits*, vol. 4, no. 6, pp. 333–342, 1969.

[55] B. Carlson, "Comparison of modern CCD and CMOS image sensor technologies and systems for low resolution imaging," in *Sensors, 2002. Proceedings of IEEE*, vol. 1, pp. 171–176 vol.1, 2002.

[56] B. Mansoorian, H.-Y. Yee, S. Huang, and E. Fossum, "A 250 mW, 60 frames/s 1280x720 pixel 9 b CMOS digital image sensor," in *Solid-State Circuits Conference, 1999. Digest of Technical Papers. ISSCC. 1999 IEEE International*, pp. 312–313, 1999.

[57] "Vision Research, Detailed Camera Comparison," *http://www.visionresearch.com.*

[58] N. Stevanovic, M. Hillebrand, B. Hosticka, U. Iurgel, and A. Teuner, "A high frame rate image sensor in standard CMOS technology," in *Solid-State Circuits Conference, 1998. ESSCIRC '98. Proceedings of the 24th European*, pp. 316–319, 1998.

[59] S. Amin-Nejad, J. Smith, and J. Lucas, "TMS320C40 parallel processor for high speed imaging application," in *Image Processing And Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465)*, vol. 1, pp. 429–434 vol.1, 1999.

[60] F. Duan, Y. Wang, and H. Liu, "A real-time machine vision system for bottle finish inspection," in *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, vol. 2, pp. 842–846 Vol. 2, 2004.

[61] T. King, M. Preston, M. Jackson, and L. Tao, "Machine vision controlled laser cutting of a non-rigid product," in *Intelligent Automation for Processing Non-Rigid Products, IEE Colloquium on*, pp. 2/1–2/4, 1994.

[62] R. White, J. Smith, and J. Lucas, "Vision-based gauge for online weld profile metrology," *Science, Measurement and Technology, IEE Proceedings* -, vol. 141, pp. 521–526, 1994.

[63] S. Amin-Nejad and J. Smith, "A visual servoing system for edge trimming of fabric embroideries by laser," *Mechatronics*, vol. 13, pp. 533–551, July 2003.

[64] J. S. Smith and C. Balfour, "Real-time top-face vision based control of weld pool size," *Industrial Robot*, vol. 32, no. 4, pp. 334–340, 2005.

[65] C. Balfour, J. Smith, and A. Al-Shamma'a, "A novel edge feature correlation algorithm for real-time computer vision-based molten weld pool measurements," *Welding Journal*, vol. 85, no. 1, pp. 1–8, 2006.

[66] L. Laiping, C. Shanben, and L. Tao, "3D vision technology and its applications in welding," in *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, vol. 1, pp. 190–198 Vol. 1, 2004.

[67] J. N. Pires, A. Loureiro, T. Godinho, P. Ferreira, B. Fernando, and J. Morgado, "Welding robots," *Robotics & Automation Magazine, IEEE*, vol. 10, pp. 45–55, 2003.

[68] Y. C. Doh, J. C. Lee, and S. H. Mun, "Camera vision system for automation of sub-assembly line in shipbuilding," in *SICE-ICASE, 2006. International Joint Conference*, pp. 5584–5586, 2006.

[69] Y.-H. Shi, G.-R. Wang, and G.-J. Li, "Adaptive robotic welding system using laser vision sensing for underwater engineering," in *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*, pp. 1213–1218, 2007.

[70] X. Liu and C. Xie, "Robotic seam tracking by utilizing arc light," in *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, vol. 2, pp. 616–621, 2006.

[71] A. Vavreck, N. Nayak, E. Bushway, and A. Ray, "An adaptive seam tracker for welding heavy-section aluminum," *Industry Applications, IEEE Transactions on*, vol. 25, pp. 658–663, 1989.

[72] P. Li and Y.-M. Zhang, "Robust sensing of arc length," *Instrumentation and Measurement, IEEE Transactions on*, vol. 50, pp. 697–704, 2001.

[73] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–165, 2004.

[74] J. M. White and G. D. Rohrer, "Image thresholding for optical character recognition and other applications requiring character image extraction," *IBM Jounral of Research and Development*, vol. 27, pp. 400–411, July 1983.

[75] K. Hashimoto, T. Kimoto, M. Kawabata, and H. Kimura, "H-infinity model-based robust control of a manipulator," in *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pp. 1597–1602 vol.3, 1991.

[76] S. Tso, P. Law, Y. Xu, and H. Shum, "Implementing model-based variable-structure controllers for robot manipulators with actuator modelling," in *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 2, pp. 701–707 vol.2, 1993.

[77] L. Rainey and M. Leahy, "Adaptive model-based control of the Utah/MIT hand," in *Systems Engineering, 1991., IEEE International Conference on*, pp. 113–116, 1991.

[78] L. Zadeh, *FUZZY SETS.* Ft. Belvoir: Defence Technical Information Centre, 1964.

[79] A. de Almeida Neto, W. R. Neto, L. Goes, and C. Nascimento, "Feedback-error-learning for controlling a flexible link," in *Neural Networks, 2000. Proceedings. Sixth Brazilian Symposium on*, pp. 273–278, 2000.

[80] G. Rovithakis and M. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 24, pp. 400–412, 1994.

[81] S. Leonard and M. Jagersand, "Learning based visual servoing," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, pp. 680–685 vol.1, 2004.

[82] J.-J. Slotine and L. Weiping, "Adaptive manipulator control: A case study," *Automatic Control, IEEE Transactions on*, vol. 33, pp. 995–1003, 1988.

[83] X. Liu and K. Fang, "A visual servoing control for robot based on fuzzy behavior and neural networks," in *Control Conference, 2006. CCC 2006. Chinese*, pp. 1629–1633, 2006.

[84] K. Stanley, J. Wu, and W. Gruver, "A hybrid neural network based vision-guided robotic system," in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, vol. 1, pp. 322–327 vol.1, 2001.

[85] R.-J. Wai and P.-C. Chen, "Robust neural-fuzzy-network control for robot manipulator including actuator dynamics," *Industrial Electronics, IEEE Transactions on*, vol. 53, pp. 1328–1349, 2006.